



EX LIBRIS
UNIVERSITATIS
ALBERTENSIS

The Bruce Peel
Special Collections
Library



Digitized by the Internet Archive
in 2025 with funding from
University of Alberta Library

<https://archive.org/details/0162020742654>

University of Alberta

Library Release Form

Name of Author: Ulrich James Zurcher

Title of Thesis: Model Based Knowledge Acquisition using Adaptive Piecewise Linear Models

Degree: Doctor of Philosophy

Year this Degree Granted: 1999

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly, or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the authors prior written permission

University of Alberta

Model Based Knowledge Acquisition using Adaptive Piecewise Linear Models

by

Ulrich James Zurcher



A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the
requirements for the degree of Doctor of Philosophy

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 1999

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Model Based Knowledge Acquisition using Adaptive Piecewise Linear Models* by *Ulrich James Zurcher* in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Two knowledge acquisition problems are addressed in this thesis: control of a pulp machine headbox and understanding of the macro contributors to pulp tensile strength. A new Computer Aided Knowledge Development technology is developed in order to effectively address these problems. The new technology, called an Adaptive Piecewise Linear Model (APLM) is an adaptive Takagi-Sugeno fuzzy system. However, rather than being interested in the extraction of knowledge from the fuzzy rules as is the usual case with fuzzy systems, this research examines the geometry of linear segments developed by the adaptation process. As the geometry of the linear segments reflects the geometry of the underlying system being modelled, the engineer is thereby able to extract significant knowledge about the process. With respect to the application to the pulp strength problem, this technology was found to offer insights into the process that had previously not been possible. A key learning was that although the process space is non-linear, the non-linearity is not as large as originally thought. Most of the data points fall on two linear segments that are at an angle of 21 degrees from each other. Although this angle is sufficiently great to preclude the use of standard linear statistical methods, it does indicate that proper control of strength should be achievable without resorting to highly elaborate control schemes.

The use of APLMs in pure modelling was also investigated. Here the control of a pulp machine headbox was compared using 3 different control approaches: traditional Model Based Predicative Control, PID Control and APLM Model Based Control. All control systems were found to be satisfactory, although the APLM control system incorporated parameter estimation in its training while this was an external step for the other approaches.

Contents

CHAPTER 1	1
1.0 INTRODUCTION	1
1.1 THESIS ORGANISATION	2
CHAPTER 2	5
2 OVERVIEW OF COMPUTER AIDED KNOWLEDGE DEVELOPMENT SYSTEMS	5
2.1 OBJECTIVES OF RESEARCH	5
2.2 CURRENT APPROACHES TO CAKD	6
2.3 ISSUES WITH CURRENT APPROACHES	20
CHAPTER 3	27
3 A NEW KNOWLEDGE DEVELOPMENT APPROACH	27
3.1 INTRODUCTION	27
3.2 COMPUTER AIDED KNOWLEDGE DEVELOPMENT USING ADAPTIVE PIECEWISE LINEAR MODELS	32
3.3 SUMMARY	56
CHAPTER 4	59

4.0 THE USE OF MATRIX DIFFERENTIATION IN THE DEVELOPMENT OF MATRIX BASED ALGORITHMS	59
4.1 INTRODUCTION	59
4.2 MATRIX OPERATIONS	60
4.2.2 THE CHAIN RULE	62
4.3 DERIVATIVES OF VECTOR AND MATRIX FUNCTIONS	63
4.4 DEVELOPMENT OF THE BACKPROPAGATION ALGORITHM USING MATRIX NOTATION	65
4.4.1 INTRODUCTION	65
4.4.2 DERIVATION OF THE BACKPROP ALGORITHM USING MATRIX NOTATION	67
4.5 COMPARISION TO NON-MATRIX NOTATION	68
4.6 SUMMARY	70

CHAPTER 5 72

5.1 MODEL BASED HEAD BOX CONTROL	72
5.2 PULP MACHINE HEADBOX CONTROL	76
5.3 MODEL BASED CONTROL OF A HEADBOX	81
5.4 PID AND APLM CONTROL OF A HEADBOX	100
5.4 A NON-LINEAR HEADBOX MODEL	105
5.5 SUMMARY	107

CHAPTER 6 109

6.0 PULP STRENGTH ANALYSIS	109
6.1 OVERVIEW	109
6.2 INTRODUCTION	110
6.3 INITIAL APLM AND NN MODELS OF PULP STRENGTH	112

6.4 APLM MODEL OF PULP STRENGTH	118
6.5 GEOMETRIC ANALYSIS OF APLM DATA	124
6.6 SUMMARY	126
<u>CHAPTER 7</u>	<u>128</u>
7.0 SUMMARY OF THESIS CONTRIBUTIONS	128
7.1 MODELLING	128
7.2 MODEL BASED HEADBOX CONTROL	129
7.3 PULP STRENGTH	130
7.4 SUMMARY	130
<u>REFERENCES</u>	<u>132</u>

Figures

FIGURE 2.1: COMPARISONS OF ERROR MEASURES	24
FIGURE 3.1: PIECEWISE LINEAR MODEL	34
FIGURE 3.2 MODEL ARTIFICIAL DATA SET	42
FIGURE 3.3 MODIFIED FIT VALUES	46
FIGURE 3.4 MAX DIFF LP MEMBERSHIP FUNCTION	50
FIGURE 3.5 AVERAGE DIFF LP MEMBERSHIP FUNCTION	50
FIGURE 3.6 CTR MEMBERSHIP FUNCTION	51
FIGURE 3.7 APLM MODEL VERSUS GENERATED DATASET	53
FIGURE 3.8 SCATTER PLOT OF APLM DATA VERSUS GENERATED DATASET	54
FIGURE 3.9 GENERATED DATA VERSUS APLM MODEL TREND PLOT	55
FIGURE 3.10 GENERATED DATA VERSUS APLM MODEL SCATTER PLOT	56
FIGURE 4.1: FEEDFORWARD NEURAL NETWORK	65
FIGURE 5.1: HEADBOX	74
FIGURE 5.2 VALVE POSITION VERSUS LEVEL BUMP TEST	80
FIGURE 5.3 FAN PUMP SPEED VERSUS HEADBOX PRESSURE	80
FIGURE 5.4: DEADBEAT UPC CONTROLLER RESPONSE TO STEP CHANGE	89
FIGURE 5.5: PROCESS OUTPUT USING MBC MIMO CONTROL	99
FIGURE 5.6 SIMULATED MACHINE RAMP-UP USING PID CONTROL	102
FIGURE 5.7 SIMULATED DE-TUNED START-UP	103
FIGURE 5.8 SIMULATED 2 STEP AHEAD CONTROL	104
FIGURE 6.1 PREDICTED VERSUS ACTUAL PULP STRENGTH	114
FIGURE 6.2 PREDICTED VERSUS ACTUAL PULP STRENGTH SCATTER PLOT	115
FIGURE 6.3 SENSITIVITY RESULTS	117
FIGURE 6.4 TREND PLOT OF MODELLED VERSUS ACTUAL STRENGTH DATA	119
FIGURE 6.5 MODELLED VERSUS ACTUAL STRENGTH SCATTER PLOT	120

FIGURE 6.6 APLM TREND PLOT FOR MODEL 2	121
FIGURE 6.7 APLM MODEL 2 SCATTER PLOT FOR MODEL 2	122
TABLE 6.2 APLM DEVELOPED LINEAR COEFFICIENTS FOR MODEL 2	122

CHAPTER 1

1.0 INTRODUCTION

Computer Aided Knowledge Acquisition is an important and burgeoning field for process and control engineers. Historically the focus of computer aided modelling has been to incorporate the engineer's mechanistic and empirical process knowledge into a model. In many cases the knowledge was incomplete or not completely accurate, however, the knowledge was sufficient for the tasks being requested of the engineer. However, the advent of powerful and inexpensive computers in the control realm has resulted in much more data being collected and retained than ever before possible. Competitive advantage is realised by those companies that are able to gain knowledge from this data to fill the gaps in their current understanding of their processes. Given the highly competitive nature of today's global economy, the engineer is being asked to provide knowledge and thereby a competitive advantage that can literally mean the survival of the company. At the same time the engineer is being asked to provide more, the number of engineers and their support staff is often being reduced. The engineer then turns to the computer systems to help fill the gaps in their knowledge.

Two knowledge acquisition problems are addressed in this thesis: control of a pulp machine headbox and understanding of the macro contributors to pulp tensile strength. A new Computer Aided Knowledge Development technology is developed in order to effectively address these problems. The new technology, called an Adaptive Piecewise Linear Model (APLM) is an adaptive Takagi-Sugeno fuzzy system. However, rather than being interested in the extraction of knowledge from the fuzzy rules as is the usual case with fuzzy systems, this research examines the geometry of linear segments developed by the adaptation process. As the geometry of the linear segments reflects the geometry of the underlying system being modelled, the engineer is thereby able to extract significant knowledge about the process. With respect to the application to the pulp strength problem, this technology was found to offer insights into the process that had previously not been possible. A key learning was that although the process space is non-linear, the non-linearity is not as large as originally thought. Most of the data points fall on two linear

segments that are at an angle of 21 degrees from each other. Although this angle is sufficiently great to preclude the use of standard linear statistical methods, it does indicate that proper control of strength should be achievable without resorting to highly elaborate control schemes.

The use of APLMs in pure modelling was also investigated. Here the control of a pulp machine headbox was compared using 3 different control approaches: traditional Model Based Predicative Control, PID Control and APLM Model Based Control. All control systems were found to be satisfactory, although the APLM control system incorporated parameter estimation in its training while this was an external step for the other approaches.

1.1 THESIS ORGANISATION

This thesis consists of the following chapters:

- **Chapter 2 - Review of current literature on the Computer Aided Knowledge Development**

This chapter outlines the industrial needs and academic significance of the research. A summary is provided of the literature reviewed in support of this research, followed by a discussion of the issues with the current state of the art, which led to the development of the new approach described in this thesis.

- **Chapter 3 – Development of a Computer Aided Knowledge Development technology**

This chapter provides the basis for the Adaptive Piecewise Linear Modelling (APLM) technology which allows the user to consider various geometric parameters as knowledge about the process being modelled. This provides a significant advantage over traditional techniques which generally operate with a black or grey box approach. With APLMs, non-linear processes are represented as a series of linear hyper-planar segments. Not only does this provide important geometric knowledge of the individual segments, but in revealing the relationship between segments' also provides quantifiable knowledge about the degree of non-linearity. This knowledge allows a process or control engineer to better design or operate their processes. Although the greatest strength of the APLM is to provide a static model of a process, APLMs can also model dynamic systems. Examples of this are shown in

this chapter along with some guidance on the issues associated with dynamic modelling using piecewise approaches.

- **Chapter 4 Development of Matrix based Differentiation Rules**

As with many engineering problems, the development of the APLM and the traditional model based headbox control system requires the differentiation of large systems of equations. Control and process engineers are familiar with the use of matrices to reduce the verbiage associated with writing the system of equations. However, when differentiation is required, the more tedious nomenclature is utilised. Using matrix differentiation allows the more concise matrix notation to be utilised throughout. Previous work on the differentiation of vectors is difficult to find and is not accompanied by any derivation of the differentiation rules. Therefore this chapter first develops the vector based rules before developing the matrix based nomenclature. As an illustrative example, the matrix based differentiation rules are applied to the development of a backpropagation algorithm for a neural network and compared to a published non-matrix based derivation.

- **Chapter 5 – Headbox Control**

In this chapter a general MIMO control law is developed and compared to an APLM developed controller. The basis for this comparison is the pulp machine headbox in Weyerhaeuser's Grande Prairie, Alberta, pulp mill. The headbox problem is a highly non-linear problem of limited dimensionality. Originally this problem was included as it was felt to be a simple application which illustrated the modelling technology and led to the more complex pulp strength problem. Upon further investigation, it was found to be the more complex of the two in many important aspects. This complexity is a result of the highly dynamic nature of the headbox. The power of the APLM modelling technique over classical techniques is well illustrated by this problem. Issues with PID control in this application are discussed with an example of the operation of an aggressively tuned controller. An APLM based model is developed of the headbox, which is then inverted to produce a controller. One of the significant advantages of linear models is the ease with which they can be inverted to produce control models. This controller is then compared to the PID control and found to offer superior control over all operating ranges.

- **Chapter 6 - Pulp Strength Modelling**

The strength of the APLM technology is in building static models of non-linear chemical processes in order to develop knowledge about the mapping between inputs and outputs. The issue of pulp strength has received significant attention from a fibre morphology standpoint. However, macro views of how pulp strength is impacted by variables in the pulp processing have received little attention. This chapter builds on previous work by this author [Zurcher 1995] where a neural network was built to predict pulp strength as measured at the end of the pulping process. The current work uses the new APLM technology to better understand the factors that impact the development of pulp strength and to determine how non-linear the problem is. Also the process in Grande Prairie has been changed since the 1995 study and the new variables are taken into account in this research.

- **Chapter 7 – Summary of Thesis Contributions**

This chapter reviews the contributions of this thesis. These include the development of the matrix based differentiation, the development of a streamlined MIMO control law, the development of the APLM technology and the development of knowledge about how pulp strength is impacted by variables in a pulp mill. Each of these developments is significant in the field of process engineering.

In summary, this thesis addresses issues related to modelling and control of chemical processes. The tools developed in this research are designed for ease of use with a specific focus on developing knowledge. Traditional tools provide significant capabilities in terms of simulation and in some cases the incorporation of a priori knowledge, however, they generally do not provide easy access to the knowledge contained in the model. If the knowledge is available, it is typically in a format that is difficult to decipher. The approaches here focus on the easy access to the computer-generated knowledge, which is increasingly becoming an important aspect of the engineer's job.

CHAPTER 2

2 OVERVIEW OF COMPUTER AIDED KNOWLEDGE DEVELOPMENT SYSTEMS

2.1 OBJECTIVES OF RESEARCH

The objective of this research was to develop knowledge about modelling pulp strength. Because this problem resides in a very non-linear, high dimensional space, the traditional modelling methodologies that have been applied to this problem have met with little success. A neural network was developed that properly mimicked the strength behaviour [Zurcher 1995], however, as is typical of neural networks, little knowledge of the underlying process could be obtained from that model. In order to develop pulp strength knowledge the following steps were required:

- ◆ Develop a thorough understanding of the current state of the art in computer aided knowledge development (CAKD) approaches. This understanding determined that CAKD technology was generally deficient in the area of model based knowledge development.
- ◆ Develop an alternative knowledge capable modelling technology using knowledge gained in step 1
- ◆ Apply the modelling technology to the pulp strength and sample problems

This chapter will address the first item, that is, the current state of the art in terms of extracting knowledge using computer learning systems.

2.2 CURRENT APPROACHES TO CAKD

2.2.1 OVERVIEW

Computer aided knowledge development is an important topic in the field of Artificial Intelligence (AI). Historically, this area has been rooted in the fields of expert systems and neural networks with some degree of rivalry between the two camps [Minsky, Papert 1969]. Lately the field of fuzzy logic has been developed and offers a third major alternative for computer aided knowledge development.

We now see several researchers in both the academic and commercial fields integrating these three technologies into robust knowledge development systems. For example, within the last several years, the Gensym company has added neural network and fuzzy logic capabilities to their G2 expert system product. As well, several of the papers discussed below combine fuzzy logic and neural networks in order to capitalise on the unique capabilities of each. The convergence of these technologies is no doubt due to the maturing of the base technologies as well as the increased computing capability available to the average user.

This research focuses on the development of knowledge pertaining to a physical process (i.e. a Kraft pulp mill). A geometric approach is used, and particular attention is given to the curvature and slope of the process response at various points in its input space. This approach will be discussed further in Chapter 3. The development of this geometric knowledge utilises fuzzy logic and therefore, fuzzy systems will be discussed in detail. Neural concepts will play a minor role. Expert systems will not be discussed in this thesis, as they offer no advantages over fuzzy logic in this application.

The development of geometric centric knowledge differs from much of the research published to date on the subject of computer aided knowledge development. The most significant difference is that most research focuses on fairly narrow areas of knowledge development whereas the geometric approach covers a broader range. In a review of relevant literature, the research with the greatest similarity was in the field of fuzzy logic rule development. Here the researchers were concerned to some degree with the geometry of the process response, but were often satisfied with simple grey box solutions.

The research on the subject of computer aided knowledge development can be classed into seven broad categories. Each of these categories illustrates approaches to development of knowledge for some particular need. Although, the methods of identifying knowledge are diverse, there are some common elements to these approaches that serve as a basis for this research.

The current research on knowledge extraction can be briefly classified as:

1. the use of developed knowledge for improved learning.
2. the development of adaptive Principle Component Analysis and Partial Least Squares.
3. system identification of an underlying process.
4. the extraction of knowledge by non-fuzzy systems.
5. the extraction of knowledge by fuzzy rule-based systems.
6. the quantification of knowledge.

Each of these classes utilises specific modelling approaches and knowledge extraction processes that are described in the following sections. The extraction of fuzzy rules will be discussed in detail while the other sections are discussed to illustrate common features among the approaches.

It will be seen in the following discussions that approaches 1 through 5 attempt to simplify the very complex task of analysing a model structure to extract some particular piece of knowledge, and, although the knowledge extraction process is simplified in order to make it possible, it is still difficult. In the case of neural based approaches this is a consequence of the neural nature of the underlying model which tends to obscure the knowledge contained in the model. The fuzzy approaches start with a much more palatable framework. However, despite this fuzzy framework, traditional non-fuzzy data processing techniques are typically utilised, which constrain the ability of these approaches to extract knowledge from a data set. The new knowledge development technology described in Chapter 3 will focus on fuzzy learning using fuzzy data handling techniques.

2.2.2 DEVELOPMENT OF IMPROVED LEARNING ALGORITHMS

When discussing computer aided knowledge development it is important to understand what type of knowledge is being developed as this typically defines the knowledge extraction methodolgy. This is especially true in the section on system identification that will be discussed later. In system identification knowledge can mean a description of the underlying process, but often simply refers to a basic simulation of the process.

In the case of developing better learning algorithms, understanding what knowledge is being extracted allows the developer to tailor the knowledge extraction process for optimum performance. Unfortunately, in much of the literature researchers often refer to developing knowledge when in fact they are only using a simple measure of the error between the model and the real system. Additionally, although there is a great deal of published research related to improved learning algorithms, much of this research is based on comparisons of the error measure using different learning algorithms without any intrinsic knowledge development. Therefore, many researchers mistakenly view knowledge as being the ability to adequately simulate a process. However, the papers in this section have been chosen because they specifically address the issue of knowledge extraction and development. These researchers attempt to use extracted knowledge (or rules) of the underlying process to improve the learning algorithm used by their learning systems.

A common theme for knowledge extraction from learning systems in general, and neural networks in particular, is that knowledge of the underlying process is easier to develop and extract if a skeletal network with a minimum number of nodes is developed (hereinafter referred to as a minimum node neural network or MNNN). All of the papers in this section attempt to achieve a minimum node count. Malinowski, *et al* [1996] discuss the use of Structured Learning with a forgetting factor to develop a MNNN. This process initially defines an overly large neural network. During training a forgetting factor is utilised to reduce weights to zero and effectively prune nonessential nodes from the network. This network is applied to the identification and control of non-linear systems and is based in part on a 1989 paper by Ishikawa. In a later paper Ishikawa [1996] updates the Structured Learning approach to utilise a selective forgetting factor. Where the previous works attempted to develop a complete set of rules with a minimum number of nodes, Ishikawa's later work attempts to perform incremental learning. The selective forgetting is designed to

allow the network to learn the most significant rules first. As learning progresses, rules of less import are learned. The advantage is that the learning process can be stopped whenever the goals of the knowledge acquisition have been met.

The above two papers use the learning algorithm to control the knowledge development. Another approach is to develop knowledge from the data before it is presented to the neural network. For example, Miyoshi, *et al* [1995] use a projection pursuit regression approach to reduce the dimensionality of the input data set. Once the dimensionality is reduced, traditional approaches can be utilised to model the process.

The key lessons from the above papers are that the development of knowledge is best achieved with a simple model structure. Complex structures become too difficult to analyse. In addition, incremental learning allows the learning process to stop at a time when sufficient knowledge has been developed to meet the goals of the knowledge extraction process. The final learning was that the structure of the input data should not be overlooked.

2.2.3 PRINCIPLE COMPONENT ANALYSIS AND PROJECTION TO LATENT STRUCTURES

Principal Component Analysis (PCA) and Projection to Latent Structures (PLS) have two key features which differentiate these approaches from the more general knowledge development approaches: first, practitioners of the PCA and PLS approaches are careful to distinguish themselves from other learning systems such as neural networks or fuzzy systems; and second, these approaches are specifically geared to the extraction of knowledge, without providing a system with which to simulate the process. The knowledge extraction approaches discussed in Section 2.2.5 typically include a simulation system as well.

Projection to Latent Structures (PLS), also known as Partial Least Squares, is a learning technique that has been the subject of much current research. Usually this technique is directly aimed at providing some knowledge about the modelled process. Li [1997] used PLS to identify primary causes of sheetbreaks on a pulp machine and Dayal [1992] investigated the primary contributors to K number variability in a Kamyr pulp digester. The strong knowledge extraction capabilities of this technique result in its primary use being in analysis (as in the above examples). However the knowledge gained should also be useful in developing

an online model of the process. This is planned as an extension to Li's work to implement on-line troubleshooting based on the PLS extracted model.

Principal Component Analysis (PCA), like PLS, is concerned with finding linear or non-linear combinations of indices that describe a set of process variables. Chessari *et al* [1995] review the use of neural networks to develop PCA using unsupervised or self-supervised learning. One of their critiques of the work to date in the area of non-linear PCA is that the analysis does not guarantee orthogonality of the indices which means that the information contained in the dataset has not been modelled with a minimum number of indices. It could be said that these analyses deliver a component analysis but not the principal component analysis. Chessari *et al* [1995] solve this problem by using a Gram-Schmidt approach in designing their neural network which his allows them to form indices that are nearly orthogonal with each other. In a test using a standard neural non-linear principal component analysis they found indices that were up to 40 degrees from orthogonal while their orthogonal non linear principal component analysis was at most less than 2 degrees from orthogonal using the same dataset.

As PCA is typically used on real world data, it must be able to cope with noisy data. There is a limit however, in terms of signal to noise ratio, over which standard PCA is acceptable. Diamantaras [1996] studied the use of PCA in noisy environments and specifically studied the robustness of Oja's [1982] basic neural PCA algorithm in these environments. Using a modified learning rule, Diamantaras was able to extract principal components in datasets that contained up to twice as much noise as could be handled by Oja's algorithm.

Most of the papers discussed in this section have focussed on PLS and PCA using neural approaches. There is a large amount of literature available on the standard non-neural use of these techniques. Li [1997] is an example of the use of a non-neural technique. The reader is also directed to this paper for a good overview of PLS and PCA.

Key learnings in this area are that improved knowledge extraction occurs when the component vectors are orthogonal. This is consistent with the conclusions drawn in the previous section as a network that provides orthogonality would also be an MNNN.

2.2.4 SYSTEM IDENTIFICATION

System Identification is the development of a suitable mathematical model from measured input and output signals and is used in the fields of control, communication and signal processing [Lo 1996]. Of particular interest here is the development and use of a process model to facilitate control of that process and System Identification has its greatest application in the areas of non-stationary and/or non-linear processes. Traditional System Identification techniques often fit sampled process data to a simple equation. Unfortunately as the actual process characteristics become more complex, or greater accuracy is required, simple equations become inadequate. Neural networks and other learning systems have achieved favour with many researchers and control engineers as these systems do not suffer from the complexity problems. Neural networks however, do suffer from such problems as slow learning, local minima in the error surface, and the inability to easily extract knowledge about the behaviour of the modelled process.

Lo [1996] researched solutions to the local minima problem by using a hybrid neural network. The neural network is broken into two portions, adaptive and non-adaptive. This network is used to solve for a plant which given appropriate initial conditions is defined by

$$y(t+1) = f(y(t), \dots, y(t-p+1), x(t), \dots, x(t-p+1), \theta(t), \varepsilon(t)) \quad 2.1$$

y are the plant outputs

x are the plant inputs

θ is a vector valued environmental parameter

ε is a vector valued environmental parameter

f is a vector valued function

Lo [1996] used a neural network to model the function and the environmental parameter. The function is modelled off-line using standard techniques incorporating *a priori* knowledge as appropriate. The environmental parameter is then learned in an adaptive manner using a neural network combined with recursive least squares or other non-neural technique. Using this combination of approaches Lo claimed superior performance when modelling moving plants. Using a simpler approach, Wang [1996] used a

neural network to model a NARMA plant. Again, by defining the structure of the NARMA model, the user provides knowledge to the modelling process.

Wang and Leu [1996] use a novel approach to modelling the stock market. Relative to industrial plants, the stock market is an interesting plant in that it is extremely noisy and highly non-stationary. In Wang and Leu's approach the input data is differenced twice so that the process becomes stationary. A recurrent neural network is then used to learn features extracted from a non-linear ARIMA model. The ability of the model to predict the Taiwanese stock market is very good. This research does not gain significant knowledge from the network, other than the black box prediction of the plant. However, the approach is interesting in that it allows the insertion of knowledge gained from the traditional modelling process into the model development. Ni, *et al* [1996] also use a neural network to identify a non-linear plant. They break the problem into two segments, a linear portion and a non-linear portion. They use their knowledge of the NARX model to develop robust recurrent neural networks that are then used to model a missile with satisfactory results.

Rahman *et al* [1997] also apply neural networks to the problem of controlling a non-linear plant. In this case, they train a neural network to learn the non-linear plant and then using this model as a basis, use another neural network to feedback linearise the system. A PI control algorithm is then used to control the linearised plant. Simulated results show very good control. In another approach to control, Iwasa *et al* [1995] use a neural network to learn the PID parameters required to control a soda and chloride plant that implements regular switches in operating conditions. Both of these papers provide examples of integrating traditional control techniques with learning systems.

One example of nonlinearity is the deadband process. Cheruel *et al* [1995] use a neural network to learn the process for use in an Internal Model Control Scheme: first that the process is modelled as a series of if-then-else rules; then the neural network is used to learn the rules. Aside from being an interesting approach, this a good example of using the minimal neural network as described in Section 2.2.2. However, as will be shown later, this type of problem may be better solved using a fuzzy system directly. In another example where fuzzy logic may be superior, Shao *et al* [1995] use a neural network to learn a feature set of machinery in terms of vibrations. Here the set of features to be identified is very well defined.

The previous three papers discuss examples of using neural networks to learn well-defined and structured knowledge. The key learning from this section is that understanding the knowledge extraction objective is important. For example, most of the researchers have used their knowledge about the traditional model structure (the knowledge objective) in developing the neural network to identify the traditional model. This learning will be applied in Chapter 3 where the geometric knowledge of a process is discussed.

2.2.5 THE EXTRACTION OF KNOWLEDGE BY A LEARNING SYSTEM

The purpose of this section is to discuss those systems that allow generalised knowledge extraction. It should be noted, however, that some systems which, in theory, allow extraction of knowledge, but in practice may be used primarily because of their capability to incorporate *a priori* knowledge which should enhance their ability to simulate the process' behaviour. In this circumstance, the systems may be used in an environment that requires this capability, (for example in a model based control scheme) rather than being used in an analytical fashion to develop an understanding of the processes being modelled.. Some fuzzy systems and Adaptive Logic Networks fall into this category. In the discussion that follows, knowledge extraction systems will be classed into rule based learning systems and non-rule based learning systems.

The most notable of the non-rule-based systems is the use of Splines (or its simpler subclass, segmented regression). Not as widely used, but very similar in concept, Adaptive Logic Networks (ALNs) are another effective non-rule-based system. ALNs and splines are both piecewise approaches with splines providing linear, quadratic, or higher order relationships, with orders of 2 or 3 being common, while ALNs are restricted to hyper-planar segments.

ALNs, developed by W. Armstrong, [Armstrong *et al* 1995; Zurcher 1994; Armstrong, Thomas 1992; Armstrong *et al* 1991] allow *a priori* knowledge to be entered prior to the training phase. This knowledge is expressed as constraints, placed by the user on the geometry of some or all of the hyper-planar segments. Training then proceeds in a manner similar to that of a neural network. Unconstrained segments learn based on the data presented to the network while the constrained segments learn to the limit of their constraints. While in theory, once an ALN is trained, the user can examine the geometry of the trained

network and use that to gain fundamental knowledge of the modelled process, research to date has not taken advantage of this capability.

Splines and segmented regression also offer the capability to incorporate knowledge into the final model. The configuration of a spline model can make significantly greater use of *a priori* knowledge than the configuration process for neural networks. The three key configuration components - knot placement, number of segments and model order - allow a flexible method for inputting knowledge. However, much of the research into splines has been aimed at providing methods for learning the optimal knot placement and number of segments for a desired accuracy. Friedman [1991] has researched this topic extensively. A method to constrain the learning algorithms in a manner similar to that used by ALNs should be possible, but has received little attention.

Rule based models recently have gained considerable popularity, with the prime examples being fuzzy models and expert systems. The ability to add *a priori* knowledge is a significant advantage over neural networks, which are generally limited to modelling those features present in the database. Few researchers have attempted to integrate *a priori* knowledge or operator experience into the neural network model. Those who have tried often resort to hybrid systems such as those made up of neural networks and fuzzy logic [Jang *et al* 1997], or fuzzy ARTMAP structures [Tan 1995]. However, when integrating *a priori* knowledge, most researchers avoid neural networks all together.

Despite the general move to hybrid structures, some researchers have successfully developed knowledge extraction from neural networks. The effectiveness of this approach is dependent on the ability to define a minimum node neural network (MNNN) where each node in the network is considered to be a rule. The work by Malinowski, *et al* [1996], Ishikawa [1996], and Miyoshi *et al* [1995] described in Section 2.2.2 and Chessari *et al* [1995] described in Section 2.2.3 rely on MNNNs. Yi and Hongbao [1995] generate a MNNN by converting the node outputs to binary, eliminating nodes that do not add significant information to the output, and then extracting knowledge (rules) from the remaining significant nodes. Huber and Berthold [1995] use a similar approach. However, instead of converting the node outputs to binary, they use a rectangular basis function network with a trapezoidal activation function. Non-significant nodes are eliminated and rules generated from the remaining nodes. Viktor *et al* [1995] reduce the network by

grouping similar nodes, setting each node in the group to the average; and, eliminating redundant nodes.

As in the previous papers, each of the remaining nodes represents a rule.

Egri and Underwood [1995] provide a different approach to knowledge extraction by learning systems.

Here a neural network is used to determine whether data from a case based reasoning system or a rule based reasoning system is valid. The data to be analysed is in the difficult field of legal reasoning. The goal was to determine whether a client's legal position was supportable in law. In this case, the simple use of rules had proven inadequate and the hybrid system was required to discriminate between two potentially different results given by the two different reasoning systems.

In all of these systems, the approach is to minimise the number of nodes that make up the network in order to maximise the knowledge contained in the remaining nodes. Clearly the extraction of knowledge is facilitated when the network is limited in size. From a different perspective, the ability of the modeller to extract knowledge is related to the ability of the model to generalise, and inversely related to the ability of the network to accurately track a specific data set. As good generalisation is a typical goal of learning systems, this capability to learn from these types of systems is very fortunate.

In this research, a minimal system has been developed. However, where most research over-specifies and then minimises, APLMs follow the example of Ishikawa [1996] to construct an under-specified network and expand it as needed.

2.2.6 EXTRACTION OF FUZZY RULES

The on-line learning of fuzzy rules has been of interest to many researchers in the last several years. Fuzzy systems provide many capabilities in terms of learning the model of a process from some data set. If the system utilises fuzzy rules then there are two additional capabilities over the "black box" neural network approach: first, *a priori* knowledge can be added through addition of new rules; and second, if a learning method is used to learn rules from data, then the developed rules may be used to generate knowledge about the modelled system.

The ability to extract knowledge from a trained fuzzy system has received much more attention from researchers. This is because fuzzy systems combine many of the advantages of neural networks and expert systems into a system that is easier for the modeller to handle. As was seen in Sections 2.2.4 and 2.2.5 special care is needed with neural networks to develop minimal structures if knowledge is to be extracted.

Minimisation improves the network's ability to generalise, which generally allows better extrapolation or interpolation on data-points that are typical of the training set but in a part of the input space not previously seen. However, the cost of increased accuracy on previously unseen data-points is a potential reduction in the accuracy against data-points that are very similar to the training set. Therefore minimal networks provide better accuracy over the entire data range at the cost of poorer accuracy over some subsets of the data range. The modeller attempting to extract knowledge from a network must correctly identify the number of neurons that balance the ability to generalise against achieving increased accuracy over some range of interest. As the number of neurons increases over the minimum, the ability to extract useful rules is reduced. Unfortunately, there are no tools to assist in achieving this balance.

Fuzzy systems, by their design, overcome this problem. Although the addition of rules likely results in the same loss of generalisation capabilities, it does not significantly reduce the modeller's ability to extract useful knowledge from a fuzzy system, as is the case when over-specifying a neural network, as due to the linguistic nature of the fuzzy rules the modeller remains able to extract the significant portions of the knowledge contained in the ruleset. This provides a key motivation to use fuzzy systems. Other reasons are widely publicised in the literature and will not be covered here.

The development of fuzzy rules is not restricted to fuzzy systems but can also be found in hybrid systems that combine fuzzy and neural techniques. Tan [1995] uses a generalisation of fuzzy ARTMAP (Cascade ARTMAP) to explicitly represent rule-based logic. Once the developer inserts *a priori* knowledge into the system, it is trained on exemplar data. Training modifies and adds to these rules. Once a system is trained, the "learned" rules can be compared to the original rules to evaluate differences between the *a priori* knowledge and the knowledge contained in the exemplars. This comparison can reveal new knowledge discovered during the training or differences between the *a priori* knowledge and "real" knowledge contained in the dataset. Nie and Lee [1995] use competitive learning to develop a fuzzy system. Similar

to the work by Lo [1996] outlined in Section 2.2.4, a two-stage approach is utilised. The first stage uses a Learning Vector Quantisation (LVQ) process to develop an initial rule set. Once training is completed, this fuzzy rule set is refined using fuzzy clustering to eliminate rules that add minimal information.

Most of the approaches previously mentioned focused on developing a minimal network. However, Mathews and Jagielska [1995] have developed two approaches to extract fuzzy rules from non-minimal networks. The first is to perform a sensitivity analysis on the hidden layer neurons to determine those neurons that have the greatest impact on the output. The user then develops rules based on the neurons that have the greatest significance. The second approach is similar however, it operates on the input nodes. The first approach is similar to a pruning mechanism (without actually effecting the pruning) for non-minimum node neural networks while the input neuron analysis extends the minimum node network approach to include additional rules. Ishibuchi and Nii [1996] take a different approach to developing fuzzy rules from a non-minimal neural network. Once the network is trained, they use the network as a simulator and use direct analysis of the inputs and outputs to build a rule base. Their process requires *a priori* classification of inputs and outputs into fuzzy classifications such as small, very small, large, etc. Although an interesting approach no discussion is made as to why, once the classification is complete, the original data set could not be used to generate the fuzzy rules and simplify the process by eliminating the neural network.

The above papers discuss the issues of learning well-structured knowledge or rule sets. However, a rule set is often not well-structured. For example the rule set

- 1) IF A THEN C
- 2) IF B THEN C
- 3) IF C THEN B

can provide a difficult problem for knowledge extraction processes due to the heterofeedback rules 2 & 3. Guoyin and Hongbao [1996] utilise a recurrent neural network to learn these types of rules. Hayashi [1989] uses a three step approach to learning fuzzy rules from a neural network where there may be conflicting rules. These steps are:

- 1) Extraction of a framework of fuzzy if-then rules
- 2) Assignment of a linguistic truth value to each rule (such as very-very-true, possibly true, etc.)
- 3) Assignment of a linguistic relative importance to each proposition

When used against a medical data set, this approach was able to learn the significant rules needed to identify diseases. Success rates of greater than 75% correct identification were reported which was over 10% better than previous efforts using other techniques.

The most complete work to date on neuro-fuzzy rule generation is provided by Jang *et al* [1997] who describe the Adaptive Neuro-Fuzzy Inference System (ANFIS) and Coactive Neuro-Fuzzy Inference System (CANFIS) systems. These systems are explicitly designed to build fuzzy rule-based systems. Although they incorporate the use of neural concepts in their approach, the majority of their effort is in the fuzzy aspects of the system.

The key learnings from the literature on fuzzy systems is that with the proper techniques, fuzzy rule based systems provide the best structure for the extraction of knowledge about a process. There are still many problems to overcome that are similar to those found in neural network approaches, however, due to their linguistic basis, the resolution of these problems is easier in the fuzzy environment.

2.2.7 ADVANCED TECHNIQUES TO DETERMINE MODEL ACCURACY

As many of the systems discussed extract knowledge from a model, it is important to have a measurement of the model's accuracy. Advanced measurement techniques permit a direct quantitative analysis of the knowledge that can be extracted from a model which can be used to determine the effectiveness of the knowledge extraction. The papers reviewed below attempt to develop knowledge measures as applied to neural networks. In general two approaches are used: the development of an uncertainty measure; or, the use of an information criterion.

As modelling is typically applied to real processes that incorporate some degree of noise, the model will only be able to predict a plant output with some degree of probability. The degree of confidence in the prediction is controlled by many factors - randomness in the real process and accuracy of the model are key

considerations. The modeller has some control over the latter, minimal control over the former and great difficulty in distinguishing between the two. Denker and LeCun [1989] use standard probability approaches to post process a neural network output to develop knowledge about the certainty in classification problems. Simoff [1996] takes the novel approach of building a network based on uncertainty using intervals. Inputs to the network are given as an interval. The network differs from standard neural networks in that it is the interval data which are acted upon as opposed to the actual values. Simoff describes some interval arithmetic that is in many ways similar to fuzzy arithmetic. Using the interval arithmetic, the nodes generate an output which also has an interval associated with it. The goal of the training process is to not only simulate the exemplar, but also to minimise the intervals. This paper illustrates the concept of uncertainty as a major component of the knowledge that we have about a plant.

The other major approach to quantifying the knowledge developed by a neural network is to determine an information criterion. This is often used to determine how well a neural network will be able to generalise as opposed to simply regurgitate the training set. In other words, the approach is to provide a measure of what type of information the network has learned. The information criteria are based on network energy or entropy. Kamimura and Nakanishi [1996] develop α information as the difference between two entropy measures. Minimising the α information allows a non-standard neural network to learn key patterns in data. Goodman, *et al* also use entropy measures to improve network performance by applying this concept to a pseudo neural network called a Connectionist Expert System. Onoda [1996] reviews several information criteria that are based on the entropy of the network.

The key learning from these papers is that there are mechanisms that can be utilised to quantify the knowledge that is learned. This is useful when determining the ability of a model to generalise, as well as to aid in the learning process. In this research there is an opportunity to use this approach to develop an information criteria for the Adaptive Piecewise Linear Model technology.

2.2.8 SUMMARY

The literature review has shown many viable approaches to the development of computer aided knowledge. Most significant is the success of the fuzzy rule based systems. For many applications the integration of

fuzzy systems with neural techniques has proven valuable. An important learning has been that several researchers have independently developed the concept of the minimum node neural network and its extension in the fuzzy field. The use of information criteria offers a mechanism to determine the ability of a system to generalise. Given the success of fuzzy systems discovered in the literature review, the fuzzy piecewise approach that is proposed in Chapter 3 will be seen to be consistent with current research, yet extending and combining the areas of fuzzy modelling and neural research. The learnings from the literature review assisted in the development of a suitable computer aided knowledge development technology to gain knowledge about the pulp strength problem.

2.3 ISSUES WITH CURRENT APPROACHES

2.3.1 INTRODUCTION

The literature review described in the previous sections show two key outages relative to Computer Aided Knowledge Extraction. First there is a lack of clear objectives for the knowledge development process. As will be discussed in Section 2.3.3, many of the systems that claim to be knowledge extraction processes are in fact only information retrieval systems. Second, the most promising CAKD systems, the fuzzy systems, eschew the fuzzy methodology for their core operations. The lack of a truly fuzzy regression process limits much of the fuzzy research. This will be discussed further in sections 2.3.4 and 2.3.5. The method described in Chapter 3 provides a solution to these outages.

2.3.2 A DEFINITION OF KNOWLEDGE

It is impossible to find a consistent definition of knowledge in the literature today. The definition that is used in this thesis is that knowledge is “the deliberate and methodical application of information to some predefined objective”. This definition is consistent with the knowledge hierarchy used by many [e.g. Rao & Qui 1993] where there is a progression from Data \Rightarrow Information \Rightarrow Knowledge \Rightarrow Intelligence (DIKI). One of the issues with the DIKI view is that the distinction between information and knowledge is typically so muddy that any clarity offered by the representation is lost. The main source of confusion is that the

difference between information and knowledge is typically context sensitive. In the definition used here, the predefined objective provides that context.

Using the above definition, one sees that the "knowledge" base of rules that may be utilised in an expert system is in reality an information base. The inference engine provides the methodical application, but knowledge does not occur until the information base and the inference engine are focused on some specific problem. In a human context, this is similar to the example of a person who has a significant "knowledge" of trivia. These people are generally not considered to be knowledgeable if all they can do is regurgitate facts. Only when they use their information base in support of some proposition do we consider that person knowledgeable. The proof of their proposition represents the predetermined objective.

2.3.3 LACK OF KNOWLEDGE IN CURRENT "KNOWLEDGE" EXTRACTION PROCESSES

Using the above definition it can be seen the much of the knowledge extraction research to date has been mere information extraction. There may be a method, but the objectives typically are defined poorly. This is especially true in the area of fuzzy rule extraction where the adaptive identification of rules is deemed important. However, what is actually done with this information, is typically considered unworthy of discussion and is not reported in the literature. In reality, the knowledge gained from the extracted information is typically more important than the information itself. Most researchers overlook this point when defining their research methodology.

This point is not just a critique that current researchers have omitted some commentary from their research summaries. The clear identification of what knowledge is to be gained (the knowledge extraction objective) is a necessary first step to extracting that knowledge. Understanding the objective will allow the researcher to identify requirements of the knowledge extraction process. It will also provide a measure to indicate whether the extraction process has been successful. Current research, which focuses on information extraction, claims success too soon.

Therefore, to qualify as a knowledge extraction system, one must be able to identify, in quantitative terms, what information will be extracted and how it will be used. Measures can then be defined to determine if

the required information has been extracted in a manner that allows it to be transformed into applicable knowledge.

2.3.4 NON-FUZZY FUZZY LEARNING

Of all the information extraction processes discussed, the adaptive development of fuzzy rules is the most promising and useful in a broad range of process modelling. Several researchers have correctly described these fuzzy models as piecewise linear systems. The methodology for fitting these linear models to non-linear systems vary, but almost all are based on a least mean squares (LMS) approach. Quite often, the outputs from these linear systems are massaged to provide fuzzy outputs. In some cases the inputs are also considered fuzzy. However, the basic cost function minimisation rarely uses a fuzzy approach. For this reason, the term non-fuzzy fuzzy learning has been coined here to indicate that although the methodologies being used include some fuzzy processes, the learning algorithm incorporates a significant non-fuzzy component -LMS - which does not provide a measure of fuzziness.

None of the research cited considers whether, or how well, a single point fits with one of the linear segments or rules being developed by the learning system. Rather a gradient descent technique, with a LMS cost function is used. The shortcomings of the LMS approach will be discussed in the next section. With proper massaging, LMS could be made to provide a pseudo fuzzy membership number, but it is non-intuitive, and none of the researchers reviewed do this transformation. Therefore, the same researchers who champion the superiority of fuzzy systems, eschew fuzziness for the very heart of their learning systems.

2.3.5 LMS COST FUNCTIONS IN FUZZY LEARNING SYSTEMS

In linear regression type models (and many fuzzy systems), the LMS cost function is given by

$$\begin{aligned} J &= \sum (y_i - \hat{y}_i)^2 \\ &= \sum (y_i - ax_i - b_i)^2 \end{aligned} \tag{2.4}$$

where

y is the set of measured values from some process

\hat{y} is the set of values produced by a model when present the set of input values x

a and b are the coefficients of the linear model.

To determine the best coefficients a and b , the error function J is minimised. Setting the derivative of J with respect to a and b to 0 and solving for a and b , results in the well known normal equations for linear regression.

Basic texts on statistic give two primary reasons for the choice of a sum of squares error function: first this results in an always-positive error; and, second, squaring the error for each point results in a heavy penalty for those points that are far from the linear approximation. The key disadvantage of using the LMS approach is its inability to provide information as to when the use of alternative methods such as segmented regression would be beneficial. As an example, Figure 2.1 illustrates the model of a piecewise linear function using linear regression. The piecewise function is given by

$$y = \begin{cases} 2x & x \leq 0 \\ 4x & x > 0 \end{cases}$$

Error Comparisons

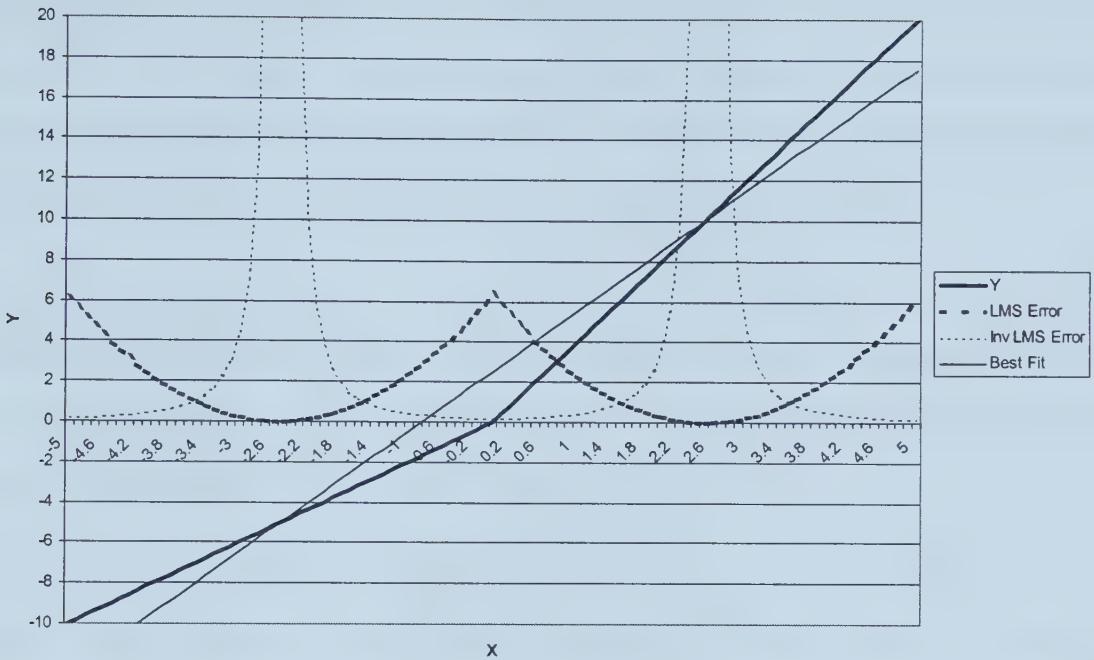


Figure 2.1: Comparisons of Error Measures

It can be seen from the figure that the linear regression plot does not provide a very good fit to the two segment base function Y . Whether the model would be satisfactory in practice would depend on the specific application. However, it readily can be seen that a two segment segmented linear regression could provide a perfect model of the base function if the knot were positioned at $x=0$. As can be seen, there is no data provided by the LMS function that would provide that assessment without the user actually looking at the graph.

There are alternative methods that could be used to identify whether segmented regression is called for. For example, if the data is available in an ordered manner, then the slope and intercept can be calculated for line segments between each pair of adjacent points. These slope and intercepts can then be sorted and evaluated for any clustering that might occur (as shown by a bimodal distribution on an histogram). If clusters exist, then segmented regression should be used. In the above example, this process would result

in two sets of slopes and intercepts and would clearly show the need for segmented regression. Of course as the dimensionality of the problem increases, this method becomes intractable and for the two-dimensional problem it is easier to just look at the graph. However, this method does show that there are automated, non-visual approaches that can be used. Again, the need for these alternative mechanisms is because the LMS approach provides little useful information about the underlying process.

The above technique identifies when segmented regression will be beneficial; however, it does not aid in the execution of that regression. The biggest problem for the user of segmented regression is the need to identify the intersections between the segments. Standard statistics texts recommend a visual identification approach, which again only works in datasets that can be visually represented.

In problems where the knot location can not be identified visually, such as high dimensional problems, an automatic method of determining the knot location is required. The existing research into splines has solved this problem of variable knot placement.

A variable knot spline is given as [Eubank 1984]:

2.5

$$y(t_i) = s(t_i) + \varepsilon(t_i) \quad i = 1, \dots, n$$

where

$$s(t_i) = \sum_{j=0}^{m-1} \alpha_j t^j + \sum_{j=0}^k \beta_j (t - \xi_j)_+^{m-1} \quad 2.6$$

with the notation

$$x_+^k = \begin{cases} x^k & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where ξ represents the knot locations. When the knot locations are free the solution requires non-linear regression [Pavlidis 1982]. Solving the non-linear regression is both complex and dependent on appropriate selection of initial values [Eubank 1984]. As well, it is often assumed that the number of knots

is known beforehand [see; for example, Chui 1988]. If the number of knots must also be determined, the problem becomes increasingly complex. An indication of the complexity is shown by the fact that, even today, trial and error is often preferred over more rigorous methods [Dierckx 1993]. Interestingly, many of the trial and error methods utilised by spline builders are similar to the pruning methods described for neural networks and fuzzy systems.

Assuming that the problem is of low enough dimensionality that the knot location can be determined without resorting to the spline techniques, the LMS approach still has some issues. The squared error provides a heavy penalty for points that are far from the linear model. This forces linear regression to identify lines that do not necessarily fit the majority of the data points, but rather attempts eliminate any major deviations. In some examples this may be appropriate; however, in others, such as modelling a chemical process, it may not. In fact, one can arguably state that the reason that the elimination of outliers has received so much attention from modellers and statisticians is because these outliers cause particular problems for the LMS algorithm.

2.3.6 SUMMARY

Although the literature review revealed many learnings on how to develop fuzzy based piecewise linear models that will permit good knowledge development, they are often restricted to low dimensional problems. As the pulp strength problem resides in a high dimensional space, new methods to develop computer aided knowledge are required. In addition the main issue with the LMS approach is that it does not inherently contain any useful knowledge about the relation between the data set and any proposed model. In the following section a fuzzy based approach to developing this knowledge is proposed to address these shortcomings.

CHAPTER 3

3 A NEW KNOWLEDGE DEVELOPMENT APPROACH

3.1 INTRODUCTION

3.1.1 CAKD STATE OF THE ART AND FUZZY SYSTEMS

Section 2.3 discussed several issues with the current state of Computer Aided Knowledge Development found when reviewing existing research. In general most of these issues related to the lack of tools available to aid the process engineer in developing knowledge about the underlying process relationships. Although one argument may be that for cases where the engineer is only attempting to build a “black box” simulator, this type of knowledge is not required, the rebuttal is that knowledge tools provide superior model verification approaches over those traditionally used by the “black box” modellers. Model verification is particularly important in neural networks as they can be over trained easily. In this circumstance, the trained neural model appears to be able to simulate a process’ outputs given a set of inputs. However closer examination reveals that the simulation is adequate only when the inputs are very similar to those used to train the model. Using inputs that require significant interpolation from the training inputs results in incorrect outputs. In other words, the network has learned the data points, but has not learned to generalise; it has not learned the underlying relationship. Hecht-Nielsen (1989) discusses the still common iterative approach to development wherein the neural network is trained on training data set, tested against a test data set and retrained as needed. Once the neural network is trained and passes the test data set, it is verified against a third dataset and the process repeated if the verification fails. Although used in other modelling fields, this approach is particularly important to the neural network field because it is one of the few methods to provide information on a network’s ability to generalise. Because of the data intensive nature of this iterative process, issues of data set integrity are raised.

A knowledge-based approach allows the engineer to more readily perform model verification. For example, if the model can be examined to determine that it has learned a linear relation between an input

and an output, and, if that matches with existing process knowledge, the engineer can be fairly confident that the model is not overtrained. Therefore, even if only a simulation is needed, it is beneficial if the engineer can perform reasonableness checks on the model that are based on an a priori process and engineering knowledge. As discussed in Chapter 2, a key component to the extraction of knowledge from a model is the configuration of the model such as MNNN or Fuzzy.

Many researchers cite the need to develop knowledge as a prime consideration for choosing fuzzy systems. However, fuzzy systems can fall short even in this area, as the knowledge described by the fuzzy rules often has a control orientation which frequently obfuscates the true underlying relationships of the system.. For example, if we were describing how to maintain a comfortable temperature in our house we might state

IF THE TEMPERATURE IS TOO COLD TURN ON THE FURNACE

IF THE TEMPERATURE IS TOO HOT TURN OFF THE FURNACE

This is an excellent description of a control process. Knowledge can be extracted to show how that the furnace has an impact on the temperature. However, if the goal were to determine how many BTUs the furnace puts out, further modelling would be required. In essence, just using a tool that is conducive to knowledge development is not sufficient to developing the desired knowledge, the proper formulation of the objective is required as well.

One way to provide this analysis is to examine the geometry of the process, or, in practice, of the model of the process. The details of the geometric concepts utilised by this research are contained in the next section, however, in general the process engineer is looking for values that indicate the degree of non-linearity within the process and how much gain (slope) exists between the inputs and the outputs. The fuzzy model can also be seen to have a geometry. The rule set given above is clearly piecewise linear. There are two regions, TOO COLD and TOO HOT. These regions are not only linear, they have zero slope, i.e. there is no distinction made between 1° too cold and 100° too cold. As well the linear regions are discontinuous in that there is no connection between TOO COLD and TOO HOT; the temperature is either one or the other. In fact, there is probably some distance between the two regions and another rule IF THE TEMPERATURE IS NEITHER TOO HOT NOR TOO COLD THEN MAKE NO CHANGES TO

FURNACE SETTING, (which is just another piecewise, zero slope discontinuous linear segment) is assumed. This is not to imply that the entire fuzzy system is discontinuous zero slope piecewise linear because the fuzzification/defuzzification processes that bracket the inference of the rules will provide the continuity and slope to the system. The entire fuzzy system will then be a non-piecewise non-linear system. However, since the fuzzification/defuzzification processes are hidden from the engineer, the visible aspects of the systems are the rules which do indicate a discontinuous zero slope piecewise linear relationship.

What knowledge could be gained from this example? First is the degree of non-linearity. As the rules show not only a linear mapping, but also a zero slope, it can be assumed that any non-linearity between the process (temperature) and the furnace is of no consequence. As well, it indicates that probably the furnace is a fixed output device. That is based on the assumption that if no attempt is made to control the heat output of the furnace, then there are likely only two states to the furnace - off or on - without any variable heat output capabilities. Therefore, while it can be deduced that the system is non-linear because of its discontinuities and has two states, in each of the states, the system can be assumed to be linear (or at least any non-linearities that exist are of little consequence). Additional data is required to obtain more knowledge about the underlying geometry of the system. For example, without some data of how long the furnace runs, it will be difficult to determine the slope of the process, or any dynamic knowledge. But just from the simple two-rule system, at least one important piece of knowledge has been developed – the degree and type of non-linearity. If the engineer is attempting to verify this control model, this knowledge could be compared to the engineer's knowledge of the process. However, this example also highlights the observation that fuzzy control systems often do not easily provide detailed knowledge about the process being modelled. This type of knowledge is often masked by any simplifications in the control rules and the fuzzification and defuzzification processes.

The above is not meant to indicate that fuzzy systems are not appropriate for computer aided knowledge development but rather as a cautionary warning that these systems must be designed and used with an understanding of their limitations. Where fuzzy systems are built using a priori knowledge, they provide excellent results. However, fuzzy systems that are designed to learn rules from a data set have some

critical issues that must be considered before their use. Still, fuzzy learning systems represent the state of the art in knowledge development tools. The engineer must remember that the knowledge developed by these systems are best suited to building control applications. To develop process knowledge, the engineer must not only evaluate the rules that were generated, but also the membership functions, the defuzzification process and the logic used by the inference engine (for example, is an AND a Zadeh, mean, or product variety?). Thus, while a complete analysis of a fuzzy system, especially a complex one, is not easy, it is superior to the knowledge analysis required in a neural network.

The model-based process analysis niche is filled by the Adaptive Piecewise Linear Modelling technology. This is focused on the development of process knowledge that will be used in the analysis of pulp strength. The downside of the tool is that after the model is built, if control is desired then the engineer must deal with membership functions, the defuzzification process and the inference engine. However, it is easier to build a fuzzy system from a knowlegebase than it is to build a knowlegebase from a fuzzy system.

The basis of the APLM technology is that process spaces can be approximately mapped into piecewise linear subspaces. This is not new and represents the approach used by segmented regression and splines. What is new is the concept of using fuzzy approaches to determine the size and orientation in space of these segments and using that geometric knowledge to analyse the process.

3.1.2 GEOMETRIC KNOWLEDGE DEVELOPMENT

The Adaptive Piecewise Linear Modelling technology takes the perspective that a process represents a functional map which transforms inputs to a single output. MIMO systems are represented by multiple functional maps, i.e., multiple MISO systems. A process output then represents a surface in a (typically) multidimensional Euclidean space. Given this perspective, the model must learn the input/output mapping. However, if the model only learns that mapping, only a simulator results. Knowledge is gained by examining the geometrical properties of the process surface at particular points of interest.

The traditional way to study these properties is to study the underlying relationships. For example, if one were to state that the jet velocity exiting a headbox is described by the equation $v^2=2gh$, then the function

surface has been described. Engineers and others by their training, know intuitively some geometric properties of the process surface, for example, always positive, parabolic, and constant curvature. In fact, this is so intuitive that most engineers perform geometric assessment without considering it to be a geometric analysis. In other words, people often progress through the information and knowledge portions of the DIKI map described in section 2.3.2 without considering the steps that are being utilised. However, geometric analysis is inherent to most engineering knowledge based analyses.

From this perspective then, the target knowledge to be developed is referred to as geometric knowledge. Geometric knowledge is felt to be particularly well suited to the functions of process and control engineers- that is, process design; process control; and process troubleshooting. The geometric parameters of interest are:

1) Maximum curvature (greatest non-linearity)

The area where curvature is the maximum represents the greatest non-linearity in the system. This would identify those operating areas that will be the most difficult to control and will signal to the operating department that either specialised control algorithms will be required or that this area of operation is to be avoided.

2) Area of maximum slope (greatest dependence)

The area of maximum slope indicates the area where one or more variables have the largest impact on the output value. This means that small changes in these manipulated variables will have a large impact on the output. Although this may be more easily controlled than the non-linear areas, this area will allow poorer control than areas of lesser slope. Due to the high dependence on the manipulated variables, increased variability in the output variable may be experienced in this operating region

3) Area of minimum slope (stable operating range)

The area of minimum slope provides a prime candidate for stable easily controlled operations. If possible the plant should run in this area.

4) Statistics of the slope and curvature for outputs versus manipulated variables.

Statistical analysis of the slope and curvature of the modelled relationships allow the process engineer to provide some overall knowledge of how the output depends on these variables. The above data will allow the engineer to make control, operating and process design decisions.

In summary, simple geometric features will be available to the modeller. These features will permit knowledge to be developed relative to the relationships between the dependent and independent variables. Using this knowledge, the modeller can design the appropriate control or operating strategies using traditional techniques.

3.2 COMPUTER AIDED KNOWLEDGE DEVELOPMENT USING ADAPTIVE PIECEWISE LINEAR MODELS

3.2.1 INTRODUCTION

This section will introduce the Adaptive Piecewise Linear Model in which fuzzy approaches are used to identify the linear segments which make up the piecewise model. The following subsections provide an overview of the technology, development of the gradient descent equations, discussion of data dependent issues, and a discussion of static versus dynamic models.

3.2.2 OVERVIEW

The goal of an APLM is to produce a piecewise linear model of a data set. Figure 3.1 shows an example of a three-piece linear representation of the sigmoid function developed by an Adaptive Piecewise Linear Model (APLM).

The inherent assumption when fitting linear approximations to a set of data is that there is some linearity present in the data. This assumption may be true only when the data are segmented into small subsets. One of the issues with standard regression using a Least Square Error (LSE) cost function is that rather than providing a measure of linearity, the LSE (Equation 2.4) is a measure of how poorly the data fit a straight line. The square term forces large excursions from the linear fit to be greatly penalised relative to small deviations. The LSE measure does not provide any indication of the degree of linearity in the data.

APLMs use a fuzzy concept of linearity. A set of points are considered to be “close” to linear with close being a fuzzy determination. As the basis is fuzzy, the measure of linearity chosen is a commonly used fuzzy membership function. The linearity measure used by APLMs is given by the Gaussian equation:

$$f = \sum e^{s(y-ax-b)^2} \quad (3.1)$$

where:

s is a linear sensitivity term and is a constant less than zero

a and b are the linear parameters of the linear segment.

This equation yields a value between 0 and 1 for each point y; one indicating that the point is on the line $ax+b$; and zero indicating that it is an infinite distance away. The linear fit measure generates a normal bell shaped curve for $-\infty < s(y-ax-b) < \infty$. Thus, points close to the line $ax+b$ will have values close to one whereas points further away will decrease in weight. A gradient descent methodology that maximises f can be used to solve for a and b.

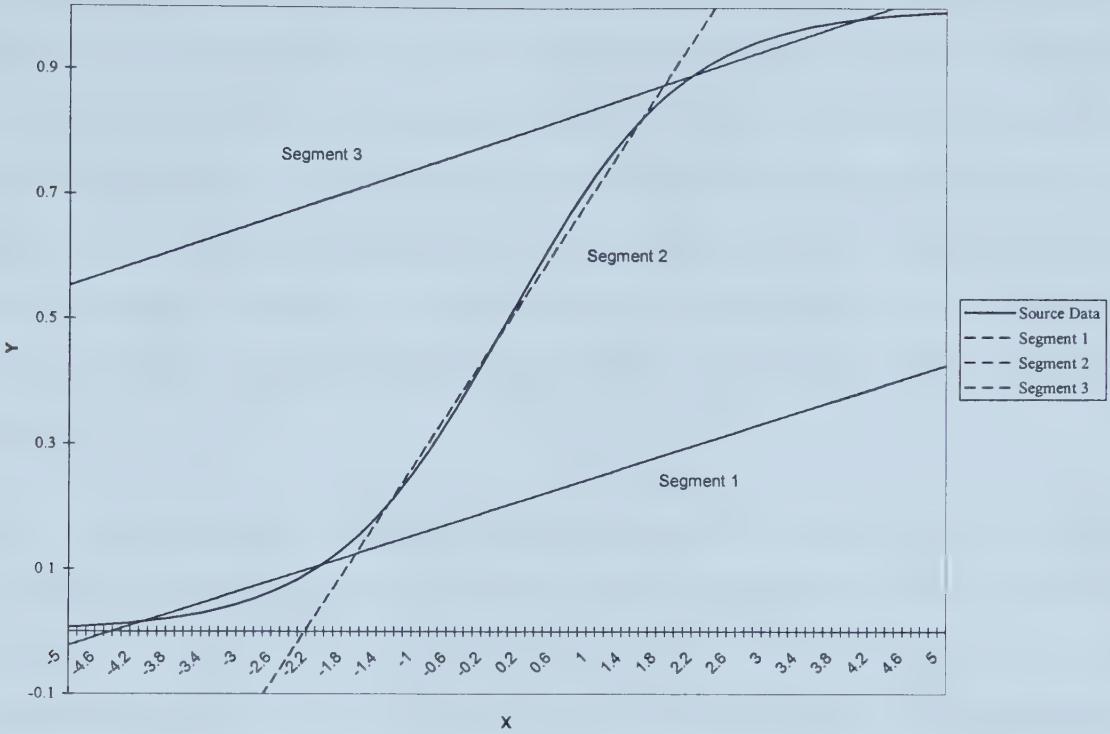


Figure 3.1: Piecewise Linear Model

The advantage of linear fit measure is that by manipulating s , different values of a and b result. As $s \rightarrow 0$, the solution for a and b approach that obtained using linear regression. As $s \rightarrow -\infty$; a and b will represent a line drawn between two points in the data-set, i.e., a perfect linear fit for a subset of two points. (For elaboration on these two points see Section 3.2.3). By adjusting s between these two extremes a solution can be found that extracts linear features from the data set. It can be seen that the impact of adjusting s is to change the sensitivity of the linear fit calculation to the error of a particular point. For the fuzzy rule, IF POINT X IS CLOSE TO LINE Y THEN..., s is the configuration parameter that determines the definition of “CLOSE”.

Initially, the fitness measure does not appear to be significantly different from the LMS approach. A small least squared error value or a large linear fit value indicates a good linear fit. But using f has two advantages. First, as f is limited to values between 0 and 1, it does not have to be normalised, whereas the

LMS value retains the units and magnitude of the underlying data. Because f is normalised, it is easily used in a computer program to determine if a point is close to a linear segment or is far away. Second, the sensitivity of f can be controlled by the appropriate choice of s . This allows the modeller to control how values that are not exactly on a linear segment are viewed, while with LMS, the calculation is fixed. This means that in the case of noisy data, the sensitivity value s can be reduced allowing a relaxed fit, whereas in the case of good data, a high value of s will more readily pick out linear features. It can be shown that modifying s changes the entropy of the system, which in turn alters the information criteria as described in Section 2.2.7.

The use of a Gaussian measure is consistent with a fuzzy approach to modelling. Along with trapezoidal and triangular equations, the Gaussian is commonly used to determine fuzzy values. The linear fit calculation will be shown to be part of a fuzzy mechanism for determining to what degree a point is part of a modelled fuzzy data set. The regression algorithm is then based on this fuzzy membership as opposed to the discrete membership used by the LMS approach.

3.2.3 EFFECT OF $S \rightarrow 0$

This section shows that as s approaches 0, the results of the fuzzy identification approach the solution determined by standard linear regression using the LMS cost function.

Examining the standard least square measure utilised by the standard linear regression formula

$$E = \sum_i (y_i - ax_i - b)^2 \quad (3.2)$$

where

E is the Squared Error

y is the actual value being modelled

a and b are the linear parameters

x is the input value

Taking the partial derivative with respect to a and b, setting it equal to zero yields

$$\frac{\partial E}{\partial a} = -2 \sum (y - ax - b)x \quad (3.3)$$

$$\frac{\partial E}{\partial b} = -2 \sum (y - ax - b) \quad (3.4)$$

Setting the respective derivatives to zero and eliminating constant values yields

$$\sum (y - ax - b)x = 0 \quad (3.5)$$

$$\sum (y - ax - b) = 0 \quad (3.6)$$

which can then be solved to yield the well known Normal Equations for Linear Regressions

The same approach can be taken with the fuzzy fit approach

$$f = \sum e^{s(y-ax-b)^2} \quad (3.7)$$

$$\frac{\partial f}{\partial a} = -2s \sum (y - ax - b)xe^{s(y-ax-b)^2} \quad (3.8)$$

$$\frac{\partial f}{\partial b} = -2s \sum (y - ax - b)e^{s(y-ax-b)^2} \quad (3.9)$$

Setting the derivatives equal to zero and eliminating constants yields

$$\sum (y - ax - b)xe^{s(y-ax-b)^2} = 0 \quad (3.10)$$

$$\sum (y - ax - b)e^{s(y-ax-b)^2} = 0 \quad (3.11)$$

it can be noted that as $s \rightarrow 0$, regardless of the values of a and b, the exponential term approaches a constant = 1 and can be eliminated. Therefore Equations 3.10 and 3.11 become (approximately)

$$\sum(y - ax - b)x = 0 \quad (3.12)$$

$$\sum(y - ax - b) = 0 \quad (3.13)$$

which are the same as equations 3.5 and 3.6 which yielded the Normal Linear Regression Equations.

3.2.4 EFFECT OF S → -∞

Examining Equations 3.10 and 3.11 it can be seen that as $s \rightarrow -\infty$, and $(y - ax - b) \neq 0$, the exponential term will approach zero.

$$\begin{aligned} \sum(y - ax - b)x &= 0 \\ 0 &= 0 \end{aligned} \quad (3.14)$$

$$\begin{aligned} \sum(y - ax - b)0 &= 0 \\ 0 &= 0 \end{aligned} \quad (3.15)$$

Therefore it will be impossible to solve for a and b . The one exception will be if in the dataset there are two points (x_n, y_n) & (x_m, y_m) such that for some a and b , $ax_n + b = y_n$ and $ax_m + b = y_m$. In that case the exponential term will become equal to 1 and equations 3.3.8 and 3.3.9 can be solved. Of course, the only possible solution will be the current values of a and b .

The consequence of this is that if there is not a good initial estimate of a and b , and if s is too large (negative), the gradient descent will not converge quickly enough to be practical. In this case the very small exponential term will cause the gradient to be too small to provide sufficient driving force for the gradient descent. If a large s is desired, then an iterative approach has proven successful. In this approach, a value of s close to -1 is chosen and a gradient descent approach used to obtain values for a and b . These values for a and b are then used as initial values for the next iteration. S is then decreased and new improved values of a and b are obtained. This process of estimating initial values for a and b is repeated until such time as the required value of s is reached. If the steps in s are small enough, this approach

ensures that there will be data points close enough to $ax+b$ that a solution can be obtained. This approach has proven successful in multidimensional problems. In two-dimensional problems, a visual estimation of a and b is also practical.

It should be emphasised that except for the solutions given in Sections 3.2.4 and 3.2.5, no analytic solutions are available for a and b . Therefore numerical approaches such as gradient descent must be used to solve for a and b . This is a detraction to the fuzzy fit approach.

3.2.5 GRADIENT DESCENT EQUATIONS

The multidimensional version of Equation 3.1 is given as

$$f = \sum e^{s(\bar{Y} - \bar{X}\bar{A}^T) \otimes (\bar{Y} - \bar{X}\bar{A}^T)} \quad (3.16)$$

To determine values for A , the derivative of the fuzzy fit function is taken with respect to the vector A yielding (see Chapter 5 for a discussion of matrix based derivatives)

$$\begin{aligned} \frac{\partial f}{\partial A} &= -2s \left(\frac{\partial}{\partial A} (\bar{Y} - \bar{X}\bar{A}^T) \right) \otimes e^{s(\bar{Y} - \bar{X}\bar{A}^T) \otimes (\bar{Y} - \bar{X}\bar{A}^T)} \\ &= -2s \frac{\partial}{\partial A} (\bar{X}\bar{A}^T) \left((\bar{Y} - \bar{X}\bar{A}^T) \otimes e^{s(\bar{Y} - \bar{X}\bar{A}^T) \otimes (\bar{Y} - \bar{X}\bar{A}^T)} \right) \\ &= -2s X^T \left((\bar{Y} - \bar{X}\bar{A}^T) \otimes e^{s(\bar{Y} - \bar{X}\bar{A}^T) \otimes (\bar{Y} - \bar{X}\bar{A}^T)} \right) \\ &= -2s X^T \left((\bar{Y} - \bar{X}\bar{A}^T) \otimes f \right) \end{aligned} \quad (3.17)$$

where

\otimes represents the Schur multiplication operation

$$e^V \equiv [e^{v_1}, e^{v_2}, \dots, e^{v_n}]^T$$

s is the sensitivity factor

Y is a $m \times 1$ column vector of y values

X is an $m \times n$ matrix of x values with the first column set to 1, i.e.

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,n} \\ 1 & x_{2,1} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \quad (3.18)$$

To solve for A, a variation of the delta learning rule is utilised

$$\overline{\mathbf{A}}_i = \overline{\mathbf{A}}_{i-1} + \overline{\ell}_i \otimes \Delta \overline{\mathbf{A}}_i \quad (3.19)$$

where

$\overline{\ell}_i$ = the current learning rate

i = the iteration in the learning process

$$\Delta \overline{\mathbf{A}}_i = \frac{\partial f}{\partial \overline{\mathbf{A}}_{i-1}} \quad (3.20)$$

$$(\ell_i)_j = \begin{cases} 1.2 * (\ell_{i-1})_j & (\Delta \mathbf{A}_i)_j * (\Delta \mathbf{A}_{i-1})_j \geq 0 \\ .1 * (\ell_{i-1})_j & (\Delta \mathbf{A}_i)_j * (\Delta \mathbf{A}_{i-1})_j < 0 \end{cases} \quad (3.21)$$

where the symbol

$(X_i)_j$ = the jth element of the vector X evaluated at iteration I

The purpose of equation 3.21 is to allow the learning rate to “grow” when the gradient descent algorithm is moving consistently in one direction, yet allow it to quickly reduce the size of its “steps” when it passes over a maximum and ΔA changes sign. Further discussion of the above can be found in most books on neural networks (e.g. Hecht-Nielsen 1989) and is offered here only as basis for the adaptation required by APLMs which will be described in Section 3.2.8.

Using equation 3.17, A can be determined using the following pseudo code:

1) Pre-process data

2) Set s = -1

- 3) For each exemplar
 - a) Obtain a new value for A using Equation 3.17
 - b) If ΔA IS NOT SMALL go to 3
- 4) Decrement s (make larger negative)
- 5) If $s > \text{threshold}_S$ go to 3
- 6) Calculate fuzzy fit values for all data points
- 7) Remove all data points from data set that meet fuzzy requirements for segment membership using rule
IF POINT IS CLOSE TO THIS LINEAR SEGMENT THEN
SAVE STATE AND REMOVE POINT FROM DATABASE
- 8) If more than threshold_P points remain, go to 2
- 9) Apply fuzzy logic point to segment allocation scheme (data post-processing)

The algorithm given in the above pseudo code incorporates the “gradual” increase of $|S|$ to ensure speedy convergence as described in the previous section. Here “gradual” is data dependent, but in practice a final value of $s=-10$ or -20 with steps of -2 or -4 work well. However, the headbox APLM described later is an example of an application that uses a final s value of -500 with a step size of -50 . The proper choice of s is application dependent, however, the choice is usually facilitated by indicators in the final model if the improper choice is made.

In Section 2.2.2 the current published research was described to show that minimal node networks are superior in the area of knowledge development. It can be seen that the APLM utilises this concept. By building up from a single segment APLM, the minimum number of nodes for a specific problem will always be found.

3.2.6 IMPLEMENTATION

The APLM modelling software is implemented as a series of Excel® macros and a support package (implemented as a DLL) written in Visual Basic®. Visual Basic® was chosen for the DLL as it allowed for quick prototyping and has some object oriented constructs. The DLL consists of two ActiveX® objects,

“vector” and “matrix” which are used extensively throughout the Excel® macros. Through the use of object oriented programming techniques and these two objects, equations such as Equations 3.16 and 3.17 are implemented directly without resorting to subscripting. This unsubscripted programming occurs at the user interface level. The objects themselves utilise subscripting where needed to perform the various vector and matrix calculations, but this is hidden from the user. For example the Schur multiplication is implemented as X.SchurMultiply(V) which is the code to implement $X \otimes V$. As well other object-oriented practices such as data hiding are employed.

The Excel macros provide an interface between the ActiveX® objects and the spreadsheets where the data is contained. A spreadsheet approach was chosen, as it reflected Weyerhaeuser’s standard development environment and allowed generation of the data necessary for program testing. Using the macros directly rather than an external program provided the ability to manipulate the data and quickly see results. The macros interact with the spreadsheet through named ranges so that the user has the utmost flexibility in setting up the APLM. A macro is provided which allows the user to specify locations for input/output data and intermediate values. The system allows the user to see the APLM converge on to the data, which provides visual confirmation that the APLM is performing as expected.

Future plans are to convert some of the macros and ActiveX® components to C++. This will rectify some performance issues with the current package, and allow greater object oriented programming. Currently Visual Basic® only supports a very limited version of object oriented programming and does not contain such capabilities as overloaded operators and functions. As well, rewriting the package will allow the package to be ported to non-Microsoft Windows® based platforms

3.2.7 DATA DEPENDENT ISSUES

One of the major problems with gradient descent techniques is the tendency for the algorithms to get trapped in a local minimum. Although the complexity of the fuzzy fit function should provide ample opportunities for local minima, in practice, when using real data this has not been seen to be a problem.

However, artificial data sets such as shown in Figure 3.1 have been generated that exhibited local minima problems.

Figure 3.2 shows a parabolic data set with an initial value for A found through linear regression. The problem with this data set is that it is perfectly symmetrical. Regardless of the value of S, there are no small changes to A that will improve the fit. Clearly a line positioned on one of the legs would be a better fit than the linear regression solution, however, there is no simple gradient descent technique that will find this solution.

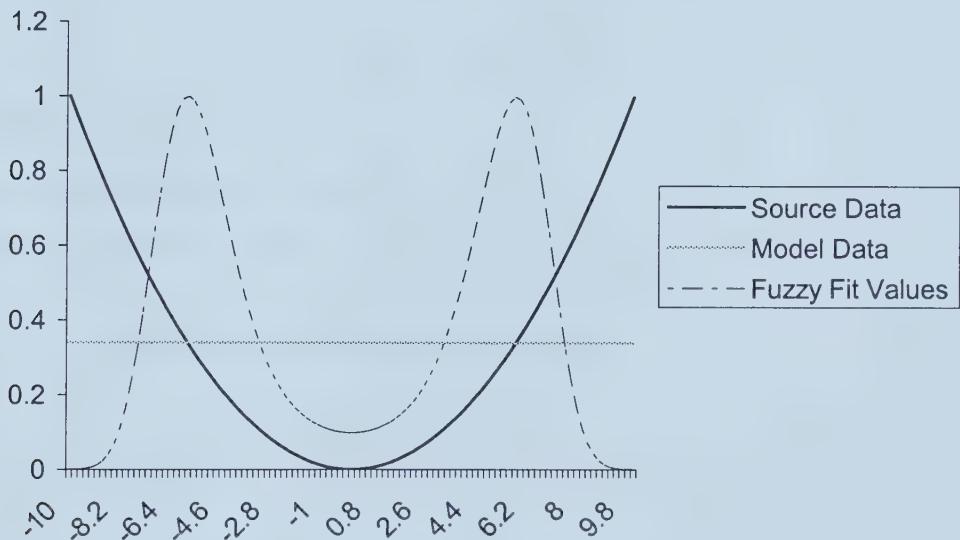


Figure 3.2 Model Artificial Data Set

Each of the parabola legs can be visualised to be a pseudo linear segment. The symmetry problem exhibited here is that there is no single route from the linear regression solution to one of the potential linear segments. One approach to this problem is to force the identification process to choose one segment by weighting the cost function to favour one segment at the expense of all others.

$$f' = e^{s(\bar{Y} - \bar{X}\bar{A}^T) \otimes (\bar{Y} - \bar{X}\bar{A}^T)} e^{s_c * edist(\bar{X}, \bar{C})} \quad (3.22)$$

where

s_c is the sensitivity value for the centre compensation

$edist$ is the square of the Euclidean distance from the centre of a linear segment and is given by

$$edist(\bar{X}, \bar{C}) \equiv \left(\left(\bar{X} - \bar{C}' \right) \otimes \left(\bar{X} - \bar{C}' \right) \right) |1| \quad (3.23)$$

where

$|1|$ is the matrix composed of all ones

C is the vector describing the centre location

C' is the matrix composed of repetitive rows of the centre position, i.e.

$$\bar{C}' \equiv \begin{bmatrix} C \\ \bar{C} \\ \vdots \\ \bar{C} \end{bmatrix} \quad (3.24)$$

Note that $edist$ yields a column vector as the effect of multiplying by the $|1|$ matrix is to sum along the rows of the $(x-c)$ matrix.

The change to the learning algorithm is restricted to the calculation of the derivative

$$\begin{aligned}
\frac{\partial f'}{\partial \mathbf{A}} &= -2s \left(\frac{\partial}{\partial \mathbf{A}} (\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \right) \otimes e^{s(\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes (\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T)} \otimes e^{s_c * \text{edist}(\bar{\mathbf{X}}, \bar{\mathbf{C}})} \\
&= -2s \frac{\partial}{\partial \mathbf{A}} (\bar{\mathbf{X}}\bar{\mathbf{A}}^T) \left((\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes e^{s(\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes (\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T)} \otimes e^{s_c * \text{edist}(\bar{\mathbf{X}}, \bar{\mathbf{C}})} \right) \\
&= -2s \mathbf{X}^T \left((\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes e^{s(\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes (\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T)} \otimes e^{s_c * \text{edist}(\bar{\mathbf{X}}, \bar{\mathbf{C}})} \right) \\
&= -2s \mathbf{X}^T \left((\bar{\mathbf{Y}} - \bar{\mathbf{X}}\bar{\mathbf{A}}^T) \otimes f' \right)
\end{aligned} \tag{3.25}$$

with all other equations in the learning algorithm remaining the same.

The determination of the segment centre is done through a modified Kohonen approach as described in the following subroutine pseudo code:

Global Data: CentreLocation(); PointCount(); NoOfCentres=0; CurrentCentre=0

- 1) Get next data point
- 2) If no centres within threshold_D distance of data point

then

create new centre, centred on the datapoint

Else

set i=number of centre that is closest to the data point

increment PointCount(i)

Update CentreLocation(i) with new datapoint.

The update is a function of the number of points in the CentreLocation and the fit value for the data point.

- 3) If fewer than threshold_P points have been processed

then

set CurrentCentre to centre with largest number of data points (PointCount(i)=max)

return CurrentCentre

else

if the centre with the most points has more than 120% of the point count held by the old current centre

then

 set CurrentCentre to centre with largest number of data points

else

 return CurrentCentre

It should be noted that as the fit factor is used in updating the centre location, and as the fit factor changes with each update of A, this subroutine is called for each data point in each iteration through the learning process.

The first test in step 2 allows the segment centre with the largest number of points to be the one chosen. However, after a significant number of points have been processed, it is assumed that the centre with the most points will have been found and the last centre is used rather than the one with the most points (which in most cases will be the same, or nearly so). In cases where there are two or more centres with similar numbers of points, this test keeps the algorithm from bouncing back and forth between centres as points in the dataset are processed. This “bouncing” continually changes the underlying error surface and renders it impossible for the gradient descent to converge. The second test in step 3 allows for a change in the current centre if changing A has caused one centre to gather significantly more data points than its competitors.

The actual subroutine as implemented is more complex as it contains code to improve robustness, such as tests to combine two centres when they overlap. As well, in practice it has been found that the centre compensation is only required when highly symmetrical data has been utilised, which has occurred only with artificial data sets. As there is a significant performance penalty exacted with this technique, the centre compensation has only been used on these data sets.

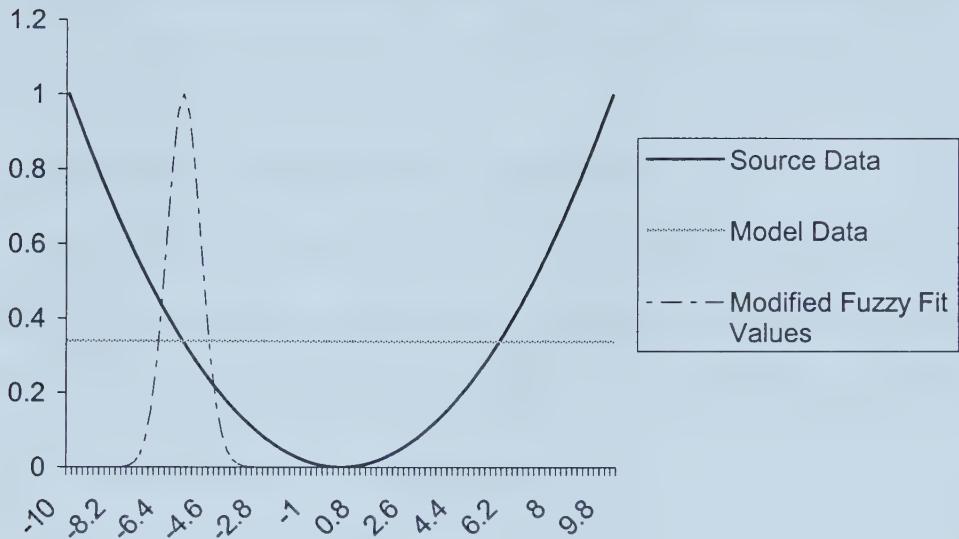


Figure 3.3 Modified Fit Values

Figure 3.3 shows the impact of adding the centre compensation to the fuzzy fit values. Here only the modelled data that is close to the left-hand leg of the parabola has high fit values. The right hand leg is ignored in this particular identification process. In this case, the gradient descent will quickly identify the left leg as a pseudo linear segment. Once the left leg is identified and removed from the dataset, the right leg will be identified on the next iteration. Other than using the modified fuzzy fit values, the pseudo code given previously applies here.

Examining Figure 3.2 it can be seen that the fuzzy fit values are a bell shaped curve centred on the point where the modelled data crosses the actual data. A comparison of Figures 3.2 and 3.3 shows that a side effect of the centre compensation is that this bell shaped curve is narrower. This serves to slow down the gradient descent process, and becomes especially noticeable for larger values of $|s|$. As s becomes increasingly negative, the impact of the centre compensation becomes greater. Therefore, the value of S_c is an inverse function of s and approaches 0 as s gets large. In this way, the impact of the centre compensation is reduced at large negative values (~ -10) of s . This approach has been successful in all tests, as the symmetry problem has only been seen to occur in linear regression type results. Once the model has

“moved off” the linear regression solutions, the need for the centre compensation to the fit factor is eliminated.

3.2.7 FUZZINESS IN THE IMPLEMENTATION OF THE APLM

The fuzzy approach utilised by the APLM technology can be illustrated by two examples in the APLM implementation. The first example is the use of the fuzzy cost function, the fuzzy gradient descent, fuzzy segment allocation rules, and associated alpha-cut threshold. The alpha-cut threshold can be either strong

$$\mu_A(x) \geq \alpha \quad (3.26)$$

or weak

$$\mu_A(x) > \alpha \quad (3.27)$$

where

$\mu_A(x)$ is the degree of membership [Cox 1999]. In APLMs a strong alpha-cut is used and as the linear segments are identified, the gradient descent mechanism is controlled by the degree of membership function. In order to provide the fastest convergence, the alpha-cut is kept very low. However, after the parameters for a linear segment have been determined, the alpha-cut is raised to a higher value, typically $>.80$. The alpha-cut value then controls how the fuzzy rule for data-point allocation is executed. This fuzzy rule is expressed as

IF POINT is CLOSE to LINEAR HYPERPLANE $f(X)$

THEN POINT is allocated to LINEAR SEGMENT X

If the centre compensation as described in Section 3.2.6 is enabled then the rule is

IF POINT is CLOSE TO LINEAR HYPERPLANE $f(X)$

AND POINT is CLOSE TO LINEAR SEGMENT X's CENTRE

THEN POINT is allocated to LINEAR SEGMENT X

The result of this adaptation process is a Takagi-Sugeno fuzzy system. In this fuzzy system the consequents of the rules are a function $b=g()$ [Passino, Yurkovich 1997]. The Takagi-Sugeno fuzzy rules for the APLM system are

IF POINT is allocated to LINEAR SEGMENT X

THEN $Y=f(Z)$

Where $f(Z)$ is the output of linear segment X at hyperpoint Z.

A more complex rule than used in adaptation of the linear segments is required when determining when to stop the adaptation process. In the case of APLMs this determination is not only used to identify when to stop training, but also to determine when to change s to a more negative value. In traditional neural networks, the decision relates only to when to stop training. The typical approach is to stop based on the error between the test set and the model. This is either done programmatically in an unattended mode, or by the user watching a plot of the error values.

For the APLM however, as only individual segments are being trained at any given time, the above measures with their whole data set perspective, are not applicable. A more valuable and specific piece of data is the effect that the training is having on the network. In the case of APLMs this is the amount of change in the linear parameters for each pass through the dataset. Therefore, at the end of each iteration three values are examined: the average change in the linear parameters; the single largest change in any one of the parameters; and, the number of iterations that have occurred since the last change in the s value. The following paragraphs describe how these values are used in a fuzzy rule to determine when to stop training or when to change the value of s.

The single largest change in any one of the parameters is the most important parameter. It is undesirable to stop training, or move on to a more negative s if there are still large changes (large being fuzzy) being made to one or more linear parameters. In the fuzzy rules described below, this parameter is referred to as diffLPMax.

The second most important parameter is the average amount of change (diffLPAvg). Even if no single parameter is being greatly changed, training should continue if there is sufficient learning activity occurring in the total network.

These two parameters cover the range of training activities in the network. The first takes a holistic view of the training process while the second looks at the network from a parameter by parameter basis.

The last piece of data is the iteration count (Ctr). Experience has shown that on occasion, even with some training activity occurring, moving on to a new s value can be accomplished without a significant training penalty. However, this is not to be done too often and is generally done when the training process appears to be taking a significant amount of time (i.e. the modeller is getting impatient). Therefore, the training is stopped only when the other parameters are acceptable, and the iteration count is high.

These three parameters are used in a set of fuzzy rules that have been developed based on training experience with APLMs. These rules can be written in a fuzzy manner as

```
IF diffLPMax is SMALL  
    AND diffLPAvg is SMALL  
    AND Ctr is ACCEPTABLE  
THEN Stop Current Training Process
```

The membership function for each variable is shown in Figures 3.4 to 3.6

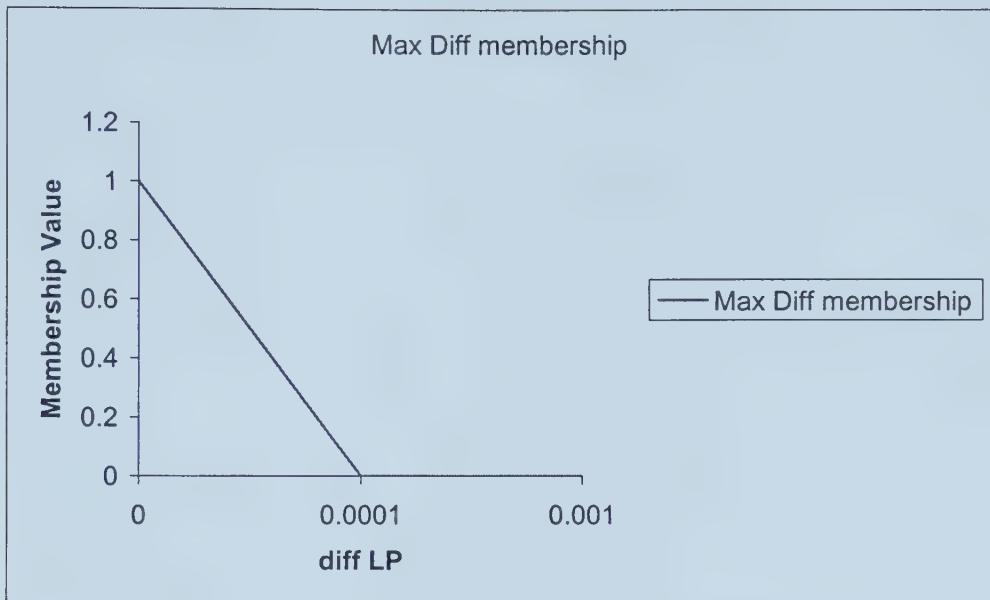


Figure 3.4 Max Diff LP Membership Function

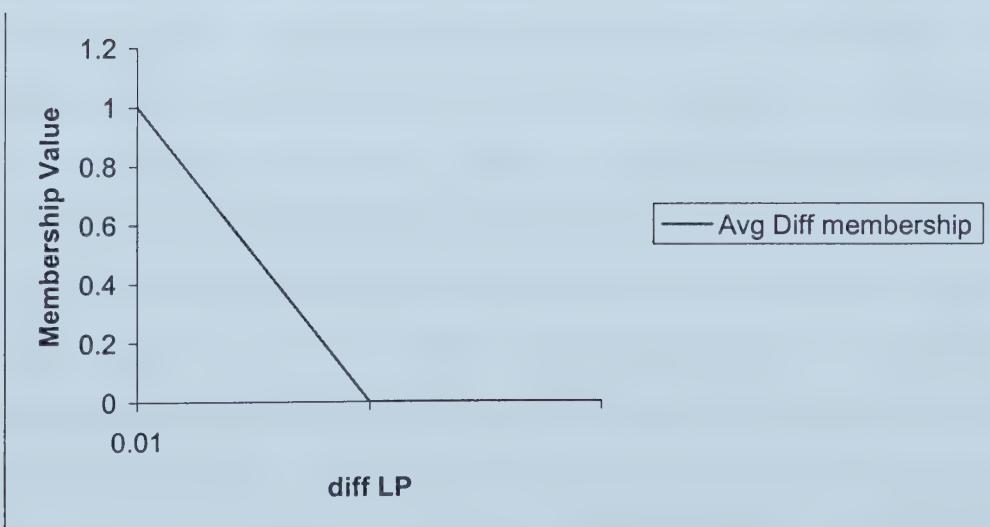


Figure 3.5 Average Diff LP Membership Function

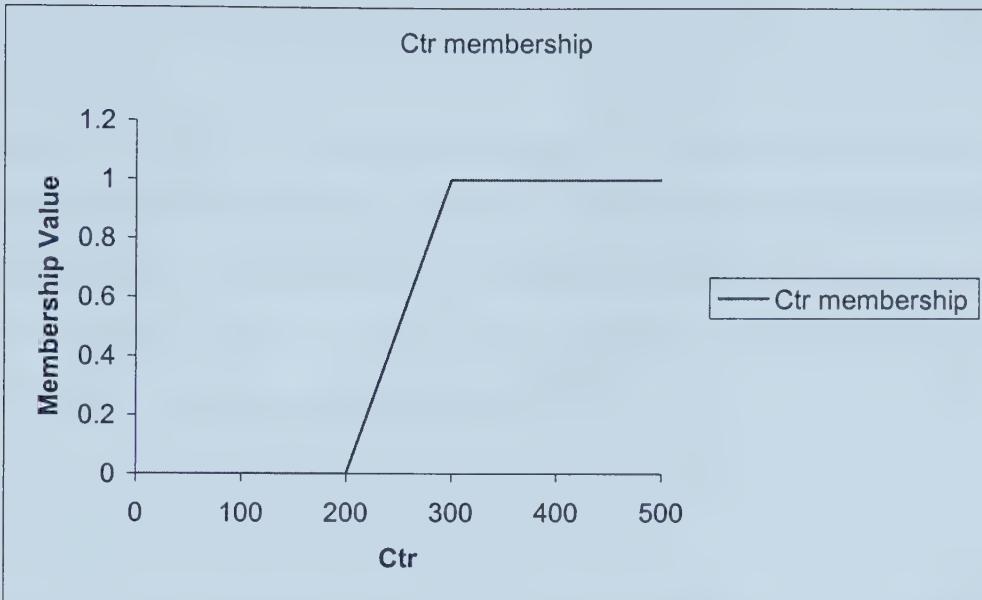


Figure 3.6 Ctr Membership Function

When describing the parameters, care was taken to identify their relative importance. Zadeh describes the fuzzy AND operator as the minimum of the two fuzzy sets [Cox 1999]. Using the rules given above, and a defuzzifier utilising a .8 threshold for true, the Zadeh AND results in training occurring until the results of each test is greater than .8. Under normal circumstances, the counter is the last value to reach that threshold which occurs when $\text{ctr}=261$. Therefore in practice, the training occurs for 261 steps before moving on to the next s or completing training. Here the fuzzy rule has almost become discrete. Several approaches have been developed to solve this type of problem (see for example [Trubatch.S, Berkan,R, 1997]), or in other words, there are approaches utilised to increase the fuzziness of the fuzzy AND. The Yager Compensatory AND operator [Cox 1998] provides one solution; however, its implementation is complex in this application. None of the other compensatory operators described in the literature address (in a straightforward manner) the situation where there is knowledge of the relative importance of the variables. Therefore, this application proposes a variation of the weighted mean AND [Cox, 1999]. This variation is defined as

$$\mu_{AND}[A, B] = (w_A \mu_A[] + w_B \mu_B[]) / (w_A + w_B) \quad (3.28)$$

Using this definition of AND the rules given above operate in a pseudo intelligent manner. If the membership values of the diffLP parameters are large (≈ 1), then the training stops sooner than if the diffLP parameters are close to .8. In this application $W_A=4$ and $W_B=1$ meaning that good performance is 4 times more important than the number of iterations. In practice, this approach has been seen to provide the similar “stop looping” decisions that a human would provide.

3.2.8 STATIC VERSUS DYNAMIC MODELS

In many chemical engineering analysis type exercises, a static model is sufficient. As a result, the prime application of APLMs is to develop these static models. However, some study into the development of dynamic models has been conducted.

As discussed above the APLM determines linear portions of data in a non-linear model. In cases where the model is already linear, and has uncorrelated inputs, the APLM will identify that linear model. Dynamic models are often described in an ARMAX form such as

$$y = \frac{\mathbf{B}}{\mathbf{A}} u + \frac{\mathbf{C}}{\mathbf{A}} \varepsilon \quad (3.29)$$

$$y(i) = a_1 y(i-1) + \dots + a_n y(i-n) + b_1 u(i-1) + \dots + b_l u(i-n) + c_i \varepsilon(i) + \dots + c_n \varepsilon(i-n)$$

It can be seen that the model is linear in its parameters and an APLM should be able to learn the model.

Consider the model

$$y(i) = .5u(i-1) + .1(y(i-1) - y(i-2)) + \varepsilon(i) \quad (3.30)$$

An APLM was trained on a data set created using the above equation. In order to provide disturbances to u , a random value of u was chosen, and held for 20 samples, and then replaced by another random value. The APLM identified the data as being generated by the equation

$$y(i) = .5069u(i-1) + .0926(y(i-1) - y(i-2)) \quad (3.31)$$

which is close to the original equation used to generate the data. Signal to noise ratio was 35db. The comparison of the APLM model results to the training set (actual data) is shown in Figure 3.7 and 3.8.

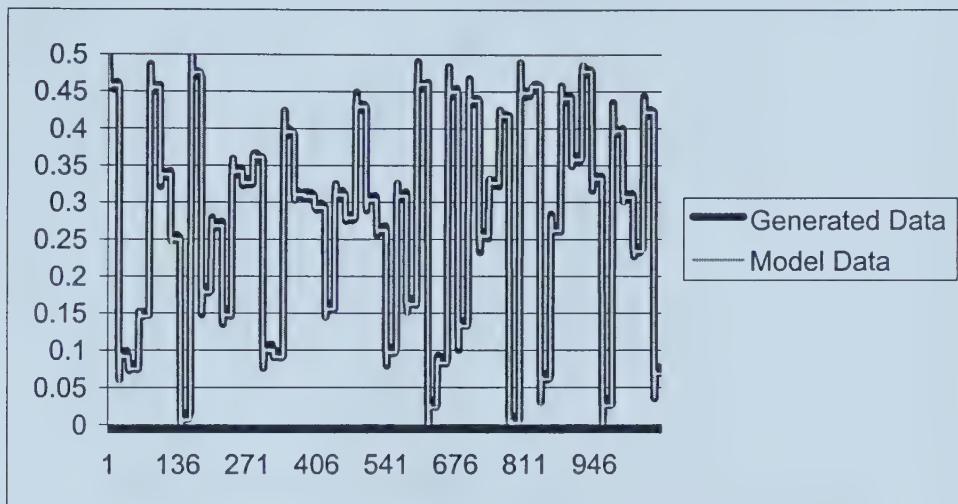


Figure 3.7 APLM Model versus Generated Dataset

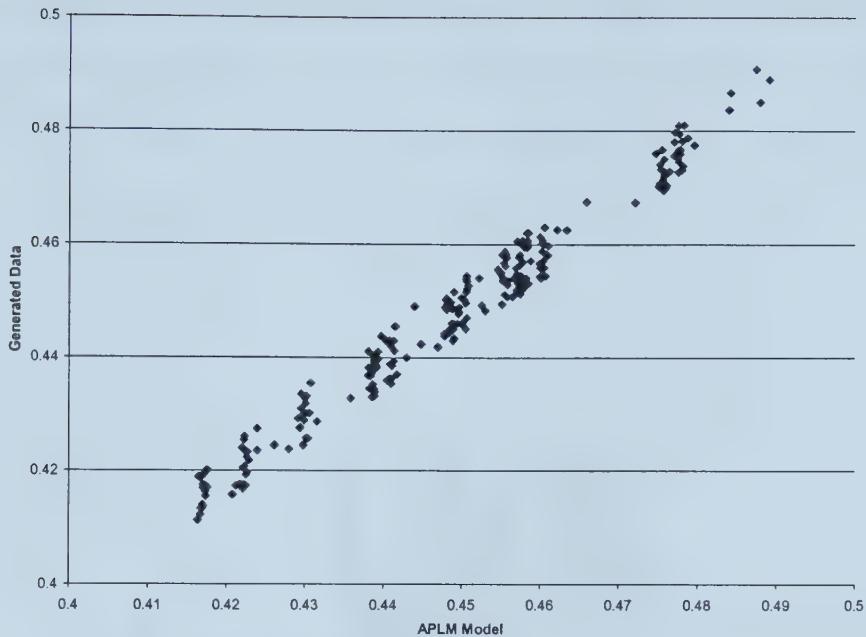


Figure 3.8 Scatter Plot of APLM Data versus Generated Dataset

One of the advantages of APLMs is their ability to automatically determine if more than one linear model is contained in a dataset. To illustrate this, a dataset was built using the formula

$$y(i) = \begin{cases} 0.5u(i-1) + 0.1(y(i-1) - y(i-2)) + \varepsilon(i) & i < 480 \\ 1.0u(i-1) + 0.3(y(i-1) - y(i-2)) + \varepsilon(i) & i \geq 480 \end{cases} \quad (3.32)$$

In this instance the amount of noise was increased to 16db for $i < 480$ and 22db for $i \geq 480$. The APLM was then trained and developed the following models

$$y(i) = \begin{cases} 0.569u(i-1) + 0.044(y(i-1) - y(i-2)) & i < 480 \\ 1.047u(i-1) + 0.288(y(i-1) - y(i-2)) & i \geq 480 \end{cases} \quad (3.33)$$

As can be seen, the APML identified the two respective models and the change in model at $i=480$. It can also be noted that the identification of the model for $i < 480$ was not as good as could be desired. The outage is assumed to be due to the significant amount of noise present. Comparing the model in Equation 3.30 and

the two models in Equation 3.33, it can be seen that as noise increases from 2% to 10% to 20%, the ability to accurately model underlying relationships from the data diminishes. There was no study done here to compare the APLM method to other methods in terms of their ability to reject noise.

The results of the two-model dataset are shown in Figures 3.9 and 3.10.

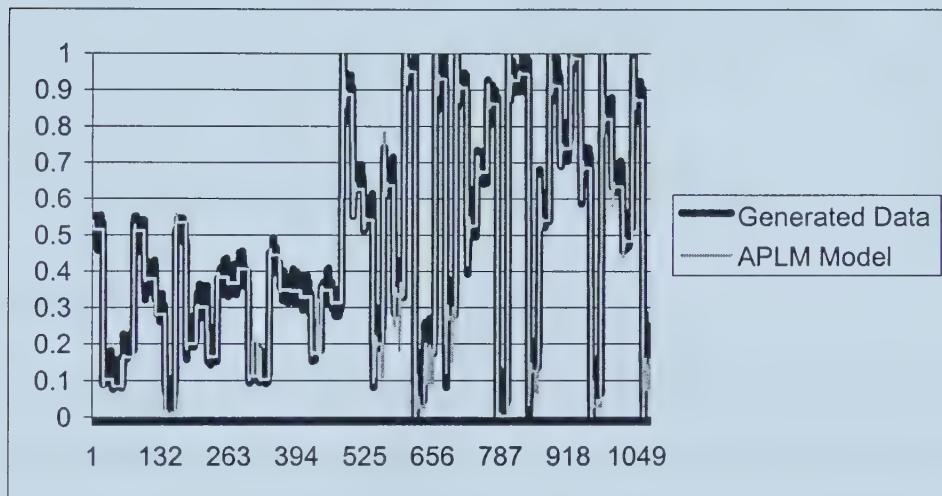


Figure 3.9 Generated Data versus APLM Model Trend Plot

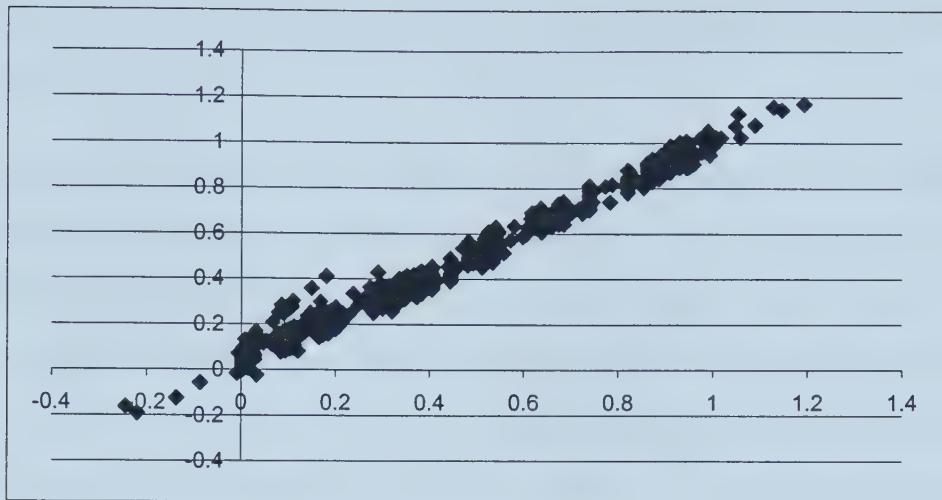


Figure 3.10 Generated Data versus APLM Model Scatter Plot

3.3 SUMMARY

The basic APLM technology was developed and applied to several artificial datasets. The implementation of the APLM was illustrated with a discussion on the impact of various configuration parameters. The effect of highly symmetrical datasets was discussed and a successful approach to analysing these types of datasets illustrated.

The ability of the APLM to accurately identify linear segments was shown, however it was noticed that the amount of noise present in the dataset had a negative impact on the APLMs ability to identify the underlying relationships. As well a discussion of dynamic versus static models was presented. It was shown that where dynamic models can be put forward in terms of linear relationships, the APLM can identify the relationships with some qualification. The qualification relates to two areas. First, as with any other adaptive learning technology, the process being modelled must have sufficient excitation. Without this excitation, there will be insufficient information in the dataset to build a knowledgeable model. Second, as will be discussed in the next chapter, the autoregressive component must not be too large. The example used in this chapter had a moderate autoregressive component of 20% to 30% of the exogenous component. The significantly larger autoregressive components used in the next chapter caused some

model mismatch. However, with sufficient excitation and moderate autoregression, the APLM is able to learn a dynamic model.

The strength of the APLM approach was seen when a mixed dynamic model was presented. In this example, the process dynamics were changed halfway through the dataset. The APLM successfully identified both models and identified when the change in model occurred (the knot). This is in contrast to standard linear regression that would have blindly provided a model that was a best fit of the data. In this case the linear regression would provide a model that reflected neither operating situation. If this had been a control example, the two segment APLM would not be practical, as there was no information provided which allowed the model to predict when the change in process occurred. The change was only identified by the change in underlying relationship and was described in terms of time from the start of the dataset. If the control engineer were unaware of the two-model situation, they would find that the liner regression model simply did not work well under any conditions. On the other hand, they would know from the information provided by the APLM that there were two process dynamics occurring and that further data was required to distinguish between these two process states. Because the traditional modelling tools do not provide knowledge about the underlying process relationships, engineers must pay greater attention to process conditions. With the APLM technology, this knowledge is an integral part of the modelling process.

The fuzziness of the internal operation of the APLM was discussed. There are two key fuzzy components to the APLM technology. First is the use of fuzzy logic to control the operation of the modelling tool itself. This is in contrast to most traditional and neural based modelling techniques that use very little knowledge in determining when a satisfactory model has been achieved. Incorporation of knowledge in knowledge development tools is not typically seen. The second component, and the most novel, is the use of a fuzzy cost function to identify those portions of a dataset that are nearly linear and so generate a pseudo Takagi-Sugeno fuzzy system. Those are key to the technology. The concept of a data point having a fuzzy membership with a linear segment is in direct contrast to most modelling techniques where the LMS cost function provides little information on linearity. In the LMS case, no information is provided to indicate

how reorientation of one or more linear segments would improve the overall performance of the model. In the case of APLMs, the fuzzy fit measure provides this type of information.

Finally, the internal workings of the APLM modelling tool were discussed as well as some data dependant issues. Most significant was the presence of highly symmetrical data that caused local minima for linear regression determined initial values. A fuzzy rule was developed that focused the model on one of the pseudo-linear segments. This rule was found to be necessary only for artificially generated datasets; however, further work may show real datasets that exhibit the same symmetry problems.

In summary the APLM technology was shown to be a viable fuzzy based modelling approach to static and dynamic applications. Key was the provision of geometric data to the modeller that provides for improved process knowledge. Further work is required in the area of dynamic processes.

CHAPTER 4

4.0 THE USE OF MATRIX DIFFERENTIATION IN THE DEVELOPMENT OF MATRIX BASED ALGORITHMS

4.1 INTRODUCTION

As seen in the previous chapter, the development of APLMs requires dealing with large systems of linear equations. Engineers are used to using matrices to ease the notation burden associated with these types of systems. However, when differentiation is required, the more burdensome notation is reverted to. This chapter will develop the rules for differentiation of a matrix with respect to another matrix. Differentiation by vectors is not a new concept and has been used by other authors (E.g. Soeterboek [1992]). However, the references for the rules associated with this technique are hard to find and only provide the result as opposed to the derivation. Also, the differentiation has been limited to vectors. Differentiation by matrices is also possible although more complex. The material that follows will review the vector differentiation and develop limited differentiation rules. Following the rule derivation, matrix differentiation will be used to develop the well-known backpropagation algorithm. In addition to being utilised in the development of APLMs, matrix differentiation rules will be applied in the next chapter when developing a model based predictive control law for the control of a coupled headbox.

The development of many multi-valued algorithms used in engineering result in several nearly identical formulae. These are often easily represented using vector or matrix notations. Often however, if differentiation is required over several variables, the more tedious notation is utilised. Differentiation of vectors and matrices with respect to other vectors and matrices is possible. Although not quite as straight forward as scalar differentials, matrix differentiation provides an easy approach to the development and proof of engineering algorithms.

Two areas where the use of matrix differentiation is of particular value are neural networks and model predictive control. Each of these topics is well suited to matrix representation. The derivation of the

Backpropagation Algorithm is well known and provides an illustrative example of the matrix-based approach. As derivatives of matrices and vectors with respect to matrices and vectors are not often used in the literature, definitions and notations will be reviewed first.

4.2 MATRIX OPERATIONS

4.2.1 DEFINITIONS

Three special matrices will be used to reduce notation when discussing derivatives of vectors and matrices. These three matrices will be denoted by a **D**, an **I**, and a numeric superscript.

Given a 1xn vector **V**, **V^D** represents the square partitioned matrix with the vector **V** along the diagonal partitions.

$$\mathbf{V}^D = \begin{bmatrix} \mathbf{V} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{V} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{V} \end{bmatrix} \quad (4.1)$$

Similarly **V^I** represents the square matrix with the individual elements of vector **V** along the diagonal.

$$\mathbf{V}^I = [v_{ij}] \quad \begin{cases} v_{ij} = v_i & i = j \\ v_{ij} = 0 & i \neq j \end{cases} \quad (4.2)$$

Finally **V^x** represents a column vector repeated x times

$$\mathbf{V}^x = [\mathbf{V} \quad \mathbf{V} \quad \mathbf{V}] = \begin{bmatrix} v_1 & v_1 & v_1 \\ v_2 & v_2 & v_2 \\ \vdots & \vdots & \vdots \\ v_p & v_p & v_p \end{bmatrix} \quad (4.3)$$

In the following definitions **e_i** is a unit vector in the **i** direction, **s** is a scalar or scalar function, **V** is a 1xp vector, **U** is a 1xm vector, and **M** is a nxm matrix.

$$\nabla_V S \equiv \begin{bmatrix} \frac{\partial s}{\partial v_1} \\ \frac{\partial s}{\partial v_2} \\ \vdots \\ \frac{\partial s}{\partial v_p} \end{bmatrix} \quad (4.4)$$

To obtain the gradient of V with respect to M, M is first vectorised by column [Brogan p 138] which will be indicated by a C superscript so that

$$\mathbf{M}^C \equiv \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1m} \\ m_{21} & m_{22} & \cdots & m_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nm} \end{bmatrix}^C \equiv \begin{bmatrix} m_{11} \\ m_{12} \\ \vdots \\ m_{1m} \\ m_{21} \\ m_{22} \\ \vdots \\ m_{2m} \\ \vdots \\ m_{n1} \\ m_{n2} \\ \vdots \\ m_{nm} \end{bmatrix} \quad (4.5)$$

Devectorising a matrix will be indicated a -c superscript, i.e.



$$\mathbf{M} = (\mathbf{M}^C)^{-C} \quad (4.6)$$

Once the \mathbf{M} is vectorised, the gradient $\overline{\overline{\nabla}}_{\mathbf{M}} \mathbf{V}$ is as given as

$$\overline{\nabla}_M V \equiv ([\nabla_M]^C)^P [V]^I = \begin{bmatrix} \frac{\partial v_1}{\partial m_{11}} & \frac{\partial v_2}{\partial m_{11}} & \dots & \frac{\partial v_p}{\partial m_{11}} \\ \frac{\partial v_1}{\partial m_{12}} & \frac{\partial v_2}{\partial m_{12}} & \dots & \frac{\partial v_p}{\partial m_{12}} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial v_1}{\partial m_{1m}} & \frac{\partial v_2}{\partial m_{1m}} & \dots & \frac{\partial v_p}{\partial m_{1m}} \\ \frac{\partial v_1}{\partial m_{21}} & \frac{\partial v_2}{\partial m_{21}} & \dots & \frac{\partial v_p}{\partial m_{21}} \\ \frac{\partial v_1}{\partial m_{22}} & \frac{\partial v_2}{\partial m_{22}} & \dots & \frac{\partial v_p}{\partial m_{22}} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial v_1}{\partial m_{2m}} & \frac{\partial v_2}{\partial m_{2m}} & \dots & \frac{\partial v_p}{\partial m_{2m}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial v_1}{\partial m_{n1}} & \frac{\partial v_2}{\partial m_{n1}} & \dots & \frac{\partial v_p}{\partial m_{n1}} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial v_1}{\partial m_{nm}} & \frac{\partial v_2}{\partial m_{nm}} & \dots & \frac{\partial v_p}{\partial m_{nm}} \end{bmatrix}^T \quad (4.7)$$

4.2.2 THE CHAIN RULE

Let $Y(M(x))$ be a vector valued function and $f(Y(M))$ a scalar function

$$\begin{aligned} \frac{df}{dm_{ij}} &= \frac{\partial y_1}{\partial m_{ij}} \frac{\partial f}{\partial y_1} + \frac{\partial y_2}{\partial m_{ij}} \frac{\partial f}{\partial y_2} + \dots + \frac{\partial y_n}{\partial m_{ij}} \frac{\partial f}{\partial y_n} \\ \overline{\nabla}_M F &= \overline{\nabla}_M Y \nabla_Y F \end{aligned} \quad (4.8)$$

4.3 DERIVATIVES OF VECTOR AND MATRIX FUNCTIONS

If \mathbf{Y} is a vector function such as

$$\mathbf{Y} = \mathbf{M}\mathbf{V}$$

then

$$\begin{aligned}\overline{\nabla}_{\mathbf{M}} \mathbf{Y} &= \overline{\nabla}_{\mathbf{M}} [\mathbf{M}\mathbf{V}] \\ &= \overline{\nabla}_{\mathbf{M}} \begin{bmatrix} m_{11}v_1 + m_{12}v_2 + \dots + m_{1n}v_n \\ m_{21}v_1 + m_{22}v_2 + \dots + m_{2n}v_n \\ \vdots \\ m_{m1}v_1 + m_{m2}v_2 + \dots + m_{mn}v_n \end{bmatrix} \\ &\neq \mathbf{V}\end{aligned}\tag{4.9}$$

(Note that when performing differentiation with matrices, particular attention must be paid to the dimensions of the resultant derivative. In the example above, $\overline{\nabla}_{\mathbf{M}} \mathbf{Y}$ is a matrix of nm rows and n columns. Therefore it is clear that \mathbf{V} can not be the resultant derivative as there are insufficient matrix elements.)

$$\overline{\nabla}_{\mathbf{M}} \mathbf{Y} = \begin{bmatrix} \frac{\partial y_1}{\partial m_{11}} & \frac{\partial y_1}{\partial m_{12}} & \dots & \frac{\partial y_1}{\partial m_{1n}} \\ \frac{\partial y_1}{\partial m_{21}} & \frac{\partial y_1}{\partial m_{22}} & \dots & \frac{\partial y_1}{\partial m_{2n}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial y_1}{\partial m_{mn}} & \frac{\partial y_1}{\partial m_{mn}} & \dots & \frac{\partial y_1}{\partial m_{mn}} \end{bmatrix}$$

$$\begin{aligned}
&= \begin{bmatrix} v_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ v_n & 0 & \cdots & 0 \\ 0 & v_1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & v_n & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & v_1 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & v_n \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{V} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{V} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{V} \end{bmatrix} \\
&= \mathbf{V}^D \tag{4.10}
\end{aligned}$$

4.4 DEVELOPMENT OF THE BACKPROPAGATION ALGORITHM USING MATRIX NOTATION

4.4.1 INTRODUCTION

As an example of the above matrix techniques the derivation of the standard backpropagation algorithm is shown. The typical non-matrix approach as described by Kosko (1992) is used for comparison purposes.

The derivation of the backpropagation algorithm assumes a feedforward neural network structure as shown in Figure 4.1.

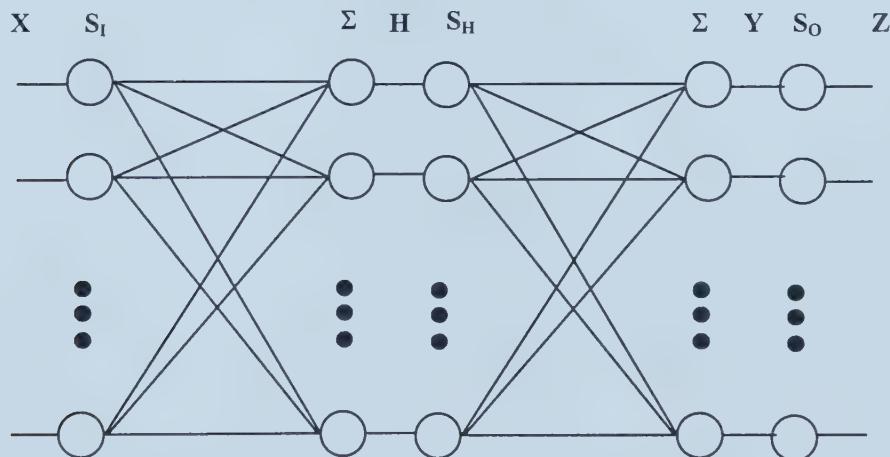


Figure 4.1: Feedforward Neural Network

Relations (from Kosko) (with E being the error, D being the desired output, m_{ij} representing the weight between the i^{th} node in the hidden layer and the j^{th} node in the output layer and n_{ij} representing the weight between the i^{th} node in the input layer and the j^{th} node in the hidden layer.

$$E = \frac{1}{2} \sum_j^p \{d_j - S_{oj}(y_j)\}^2 \quad (4.11)$$

$$y_j = \sum_q S_{Hq}(h_q) m_{qj} \quad (4.12)$$

$$\begin{aligned} h_q &= \sum_i S_h(x_i) n_{iq} \\ &= \sum_i x_i n_{iq} \end{aligned} \quad (4.13)$$

if the transfer function of the input layer is unity.

From Kosko (p208), the weight changes between the last hidden layer and the output layer are given by

$$\begin{aligned} \Delta m_{qj} &= -\frac{\partial E}{\partial m_{qj}} \\ &= -\frac{\partial E}{\partial y_j} \frac{dy_j}{dm_{qj}} \\ &= -\frac{\partial E}{\partial y_j} S_{Hq}(h_q) \\ &= -\frac{\partial E}{\partial S_j(y_j)} \frac{\partial S_j(y_j)}{\partial y_j} S_{Hq}(h_q) \\ &= -[d_j - S_j(y_j)] \frac{\partial S_j(y_j)}{\partial y_j} S_{Hq}(h_q) \\ &= -\delta_j \partial S'_j S_{Hq}(h_q) \end{aligned} \quad (4.14)$$

where

$$\delta_j = d_j - S_j(y_j)$$

$$S'_j = \frac{\partial S_j(y_j)}{\partial y_j}$$

In vector notation, the change in weights for the entire matrix M can be expressed as

$$\begin{aligned}
\Delta \mathbf{M} &= -[\Delta m_{qj}] \\
&= -[\delta_j S'_j S_q(h_q)] \\
&= -\begin{bmatrix} \delta_1 S'_1 S_1(h_1) & \delta_2 S'_2 S_1(h_1) & \cdots & \delta_m S'_m S_1(h_1) \\ \delta_1 S'_1 S_2(h_2) & \delta_2 S'_2 S_2(h_2) & \cdots & \delta_m S'_m S_2(h_2) \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1 S'_1 S_n(h_n) & \delta_2 S'_2 S_n(h_n) & \cdots & \delta_m S'_m S_n(h_n) \end{bmatrix} \tag{4.15}
\end{aligned}$$

4.4.2 DERIVATION OF THE BACKPROP ALGORITHM USING MATRIX NOTATION

The matrix representations of equations 4.11-4.13 are

$$E = \frac{1}{2} [\mathbf{D} - \mathbf{S}_o(\mathbf{Y})]^T [\mathbf{D} - \mathbf{S}_o(\mathbf{Y})] \tag{4.16}$$

$$\mathbf{Y} = \mathbf{MS}_H(\mathbf{H}) \tag{4.17}$$

$$\mathbf{H} = \mathbf{NS}_I(\mathbf{X}) \tag{4.18}$$

$$\begin{aligned}
\Delta \mathbf{M} &= -\overline{\nabla}_{\mathbf{M}} \mathbf{E} \\
&= -\overline{\nabla}_{\mathbf{M}} \mathbf{Y} \nabla_{\mathbf{Y}} \mathbf{E} \\
&= -\left(\mathbf{S}_H(\mathbf{H})^D \nabla_{\mathbf{Y}} \mathbf{E} \right)^T \\
&= -\left(\mathbf{S}_H(\mathbf{H})^D (\nabla_{\mathbf{Y}} \mathbf{S}_o(\mathbf{Y}))^I \nabla_{\mathbf{S}_o(\mathbf{Y})} \mathbf{E} \right)^T \\
&= -\left(\mathbf{S}_H(\mathbf{H})^D (\nabla_{\mathbf{Y}} \mathbf{S}_o(\mathbf{Y}))^I [\mathbf{D} - \mathbf{S}_o(\mathbf{Y})] \right)^T \tag{4.19}
\end{aligned}$$

Expanding 4.19 provides the same result as given in 4.15.

The weight changes for the hidden layer are given by

$$\begin{aligned}
\Delta \mathbf{N} &= -\overline{\overline{\nabla}}_{\mathbf{N}} \mathbf{E} \\
&= -\overline{\overline{\nabla}}_{\mathbf{N}} \mathbf{H} \nabla_{\mathbf{H}} \mathbf{E} \\
&= -\left(\mathbf{S}_I(\mathbf{X})^D \nabla_{\mathbf{H}} \mathbf{E} \right)^T \\
&= -\left(\mathbf{S}_I(\mathbf{X})^D \left((\nabla_{\mathbf{H}} \mathbf{S}_H(\mathbf{H}))^I \nabla_{\mathbf{S}_H(\mathbf{H})} \mathbf{E} \right) \right)^T \\
&= -\left(\mathbf{S}_I(\mathbf{X})^D \left((\nabla_{\mathbf{H}} \mathbf{S}_H(\mathbf{H}))^I \left((\nabla_{\mathbf{S}_H(\mathbf{H})} \mathbf{Y}) \nabla_{\mathbf{Y}} \mathbf{E} \right) \right) \right)^T \\
&= -\left(\mathbf{S}_I(\mathbf{X})^D \left((\nabla_{\mathbf{H}} \mathbf{S}_H(\mathbf{H}))^I \left((\nabla_{\mathbf{S}_H(\mathbf{H})} \mathbf{Y}) (\nabla_{\mathbf{Y}} \mathbf{S}_O(\mathbf{Y}) [\mathbf{D} - \mathbf{S}_O(\mathbf{Y})]) \right) \right) \right)^T
\end{aligned} \tag{4.20}$$

which when expanded yields the same result as given by Kosko on Page 209.

4.5 COMPARISION TO NON-MATRIX NOTATION

In this section it is shown that the expansion of matrix notation agrees with the non-matrix based backpropagation results as developed by Kosko on page 207. The expansions will assume a neural network with 3 input nodes, 3 hidden nodes and 3 output nodes. Transfer functions in each node are not assumed to be the same although the transfer function at the input layer is assumed to be unity.

4.5.1 WEIGHT CHANGES BETWEEN HIDDEN AND OUTPUT LAYERS

$$\begin{aligned}
 \Delta \mathbf{M} &= -\left(\mathbf{S}_H(\mathbf{H})^D (\nabla_{\mathbf{Y}} \mathbf{S}_O(\mathbf{Y}))^T [\mathbf{D} - \mathbf{S}_O(\mathbf{Y})] \right)^T \\
 &= -\left(\begin{bmatrix} S_1(h_1) & 0 & 0 \\ S_2(h_2) & 0 & 0 \\ S_3(h_3) & 0 & 0 \\ 0 & S_1(h_1) & 0 \\ 0 & S_2(h_2) & 0 \\ 0 & S_3(h_3) & 0 \\ 0 & 0 & S_1(h_1) \\ 0 & 0 & S_2(h_2) \\ 0 & 0 & S_3(h_3) \end{bmatrix} \begin{bmatrix} S'_1 & 0 & 0 \\ 0 & S'_2 & 0 \\ 0 & 0 & S'_3 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \right)^T \\
 &= -\left(\begin{bmatrix} S_1(h_1)S'_1 & 0 & 0 \\ S_2(h_2)S'_1 & 0 & 0 \\ S_3(h_3)S'_1 & 0 & 0 \\ 0 & S_1(h_1)S'_2 & 0 \\ 0 & S_2(h_2)S'_2 & 0 \\ 0 & S_3(h_3)S'_2 & 0 \\ 0 & 0 & S_1(h_1)S'_3 \\ 0 & 0 & S_2(h_2)S'_3 \\ 0 & 0 & S_3(h_3)S'_3 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \right)^T = \begin{bmatrix} S_1(h_1)S'_1\delta_1 \\ S_2(h_2)S'_1\delta_1 \\ S_3(h_3)S'_1\delta_1 \\ S_1(h_1)S'_2\delta_2 \\ S_2(h_2)S'_2\delta_2 \\ S_3(h_3)S'_2\delta_2 \\ S_1(h_1)S'_3\delta_3 \\ S_2(h_2)S'_3\delta_3 \\ S_3(h_3)S'_3\delta_3 \end{bmatrix} \\
 &= \begin{bmatrix} S_1(h_1)S'_1\delta_1 & S_1(h_1)S'_2\delta_2 & S_1(h_1)S'_3\delta_3 \\ S_2(h_2)S'_1\delta_1 & S_2(h_2)S'_2\delta_2 & S_2(h_2)S'_3\delta_3 \\ S_3(h_3)S'_1\delta_1 & S_3(h_3)S'_2\delta_2 & S_3(h_3)S'_3\delta_3 \end{bmatrix} \quad \bullet
 \end{aligned} \tag{4.21}$$

This is the same result as in equation 14 or page 208 of Kosko

4.5.2 WEIGHT CHANGES BETWEEN INPUT AND HIDDEN LAYER.

$$\begin{aligned}
\Delta \mathbf{N} &= - \left(\mathbf{S}_I(\mathbf{X})^D \left(\nabla_{\mathbf{H}} \mathbf{S}_H(\mathbf{H})^I \left(\nabla_{\mathbf{S}_H(\mathbf{H})} \mathbf{Y} \left(\nabla_{\mathbf{Y}} \mathbf{S}_O(\mathbf{Y}) [\mathbf{D} - \mathbf{S}_O(\mathbf{Y})] \right) \right) \right)^T \right. \\
&= - \left[\begin{array}{ccc|ccc|c} x_1 & 0 & 0 & S'_{h1} & 0 & 0 & S'_{o1} & 0 & 0 \\ x_2 & 0 & 0 & 0 & S'_{h2} & 0 & 0 & S'_{o2} & 0 \\ x_3 & 0 & 0 & 0 & 0 & S'_{h3} & 0 & 0 & S'_{o3} \end{array} \mathbf{M} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \right]^T \\
&= - \left[\begin{array}{ccc|ccc|c} x_1 S'_{h1} m_{11} S'_{o1} & x_1 S'_{h1} m_{12} S'_{o2} & x_1 S'_{h1} m_{13} S'_{o3} & x_2 S'_{h1} m_{11} S'_{o1} & x_2 S'_{h1} m_{12} S'_{o2} & x_2 S'_{h1} m_{13} S'_{o3} & x_3 S'_{h1} m_{11} S'_{o1} & x_3 S'_{h1} m_{12} S'_{o2} & x_3 S'_{h1} m_{13} S'_{o3} \\ x_2 S'_{h1} m_{11} S'_{o1} & x_2 S'_{h1} m_{12} S'_{o2} & x_2 S'_{h1} m_{13} S'_{o3} & x_1 S'_{h2} m_{21} S'_{o1} & x_1 S'_{h2} m_{22} S'_{o2} & x_1 S'_{h2} m_{23} S'_{o3} & x_2 S'_{h2} m_{21} S'_{o1} & x_2 S'_{h2} m_{22} S'_{o2} & x_2 S'_{h2} m_{23} S'_{o3} \\ x_3 S'_{h1} m_{11} S'_{o1} & x_3 S'_{h1} m_{12} S'_{o2} & x_3 S'_{h1} m_{13} S'_{o3} & x_1 S'_{h2} m_{21} S'_{o1} & x_1 S'_{h2} m_{22} S'_{o2} & x_1 S'_{h2} m_{23} S'_{o3} & x_2 S'_{h2} m_{21} S'_{o1} & x_2 S'_{h2} m_{22} S'_{o2} & x_2 S'_{h2} m_{23} S'_{o3} \\ x_1 S'_{h2} m_{21} S'_{o1} & x_1 S'_{h2} m_{22} S'_{o2} & x_1 S'_{h2} m_{23} S'_{o3} & x_3 S'_{h2} m_{21} S'_{o1} & x_3 S'_{h2} m_{22} S'_{o2} & x_3 S'_{h2} m_{23} S'_{o3} & x_1 S'_{h3} m_{31} S'_{o1} & x_1 S'_{h3} m_{32} S'_{o2} & x_1 S'_{h3} m_{33} S'_{o3} \\ x_2 S'_{h2} m_{21} S'_{o1} & x_2 S'_{h2} m_{22} S'_{o2} & x_2 S'_{h2} m_{23} S'_{o3} & x_3 S'_{h2} m_{21} S'_{o1} & x_3 S'_{h2} m_{22} S'_{o2} & x_3 S'_{h2} m_{23} S'_{o3} & x_2 S'_{h3} m_{31} S'_{o1} & x_2 S'_{h3} m_{32} S'_{o2} & x_2 S'_{h3} m_{33} S'_{o3} \\ x_3 S'_{h2} m_{21} S'_{o1} & x_3 S'_{h2} m_{22} S'_{o2} & x_3 S'_{h2} m_{23} S'_{o3} & x_1 S'_{h3} m_{31} S'_{o1} & x_1 S'_{h3} m_{32} S'_{o2} & x_1 S'_{h3} m_{33} S'_{o3} & x_3 S'_{h3} m_{31} S'_{o1} & x_3 S'_{h3} m_{32} S'_{o2} & x_3 S'_{h3} m_{33} S'_{o3} \end{array} \right]^T \\
&= - \begin{bmatrix} x_1 S'_{h1} \sum(m_{1i} S'_{oi} \delta_i) & x_1 S'_{h2} \sum(m_{2i} S'_{oi} \delta_i) & x_1 S'_{h3} \sum(m_{3i} S'_{oi} \delta_i) \\ x_2 S'_{h1} \sum(m_{1i} S'_{oi} \delta_i) & x_2 S'_{h2} \sum(m_{2i} S'_{oi} \delta_i) & x_2 S'_{h3} \sum(m_{3i} S'_{oi} \delta_i) \\ x_3 S'_{h1} \sum(m_{1i} S'_{oi} \delta_i) & x_3 S'_{h2} \sum(m_{2i} S'_{oi} \delta_i) & x_3 S'_{h3} \sum(m_{3i} S'_{oi} \delta_i) \end{bmatrix} \quad (4.22)
\end{aligned}$$

4.6 SUMMARY

The use of matrix differentiation is a technique that is not often used by control engineers. In fact, based on interviews with several control engineers and mathematics professors, many did not know that the approach was even possible. This is likely due to the lack of publications on the technique and the difficulty in visualising and using the techniques. In many ways the techniques are highly intuitive as they follow the general rules associated with scalar differentiation. However, the user must at all times be conscious of the

underlying matrix rules. Therefore, some non-scalar constructs are developed that may lead the user astray. By showing the derivation for these matrix-based rules, it is hoped that the rules will make more sense to engineers and the pitfalls that trap the normal user can be avoided.

The use of these techniques can greatly speed the development of algorithms in the control and neural fields. As an example, the backpropagation algorithm was compared to the traditional development. This not only showed that the matrix techniques developed the same answers, but showed that the development was cleaner and more readily understood. The techniques were used in this thesis in the development of the APLM and will be used in the next chapter in the development of a general MIMO control law.

CHAPTER 5

5.1 MODEL BASED HEAD BOX CONTROL

5.1.1 INTRODUCTION

The purpose of this chapter is to develop a model based predictive control for an air-padded pulp machine headbox. Two approaches will be taken. First a traditional Unified Model-based Predictive Control scheme will be developed using techniques developed by Soeterboeck [1992]. The performance of this scheme, then will be compared to a traditional PID control and one developed using APLM techniques. The data used to develop these models was first published in Zurcher *et al* [1995]. Non-linear control approaches are not considered here, as many of the pulp and paper sites prefer to use linear methods unless absolutely necessary due to lack of control skills in the maintenance organisations.

5.1.2 HEADBOX OPERATIONAL OVERVIEW

The purpose of a pulp machine is to take the pulp product as produced by the digesting and bleach plants and convert it into a form that can be cheaply transported to the end user, typically a papermaker. In this case, a papermaker is a generic term and can also refer to tissue and towel makers. In the process being studied, this conversion will consist of forming the pulp into a continuous sheet weighing about 480 g/m^2 and 10% moisture content. As the end user will reconvert the pulp to a slurry for use in their process, the finished product specifications in terms of weight and moisture are not as stringent as for a paper sheet. However, due to the extreme weight and the stresses involved during the drying process, large deviations in weight and moisture content will cause poor reliability of the machine. This reliability will either be seen as sheetbreaks, which have significant production and quality impacts, or reduced rate operation. Because of this reliability issue, the operations group have imposed their own moisture and ream weight targets. There are targets for both the machine direction and the cross machine direction profiles.

The pulp machine consists of a headbox that delivers a layer of low consistency slurry onto a moving endless screen (wire) which is about 10 meters long. The purpose of the wire is to draw water from the

slurry using gravity and vacuum until the slurry becomes a sheet of about 20% pulp (by weight). The sheet is then pressed in the press section that uses mechanical force to dewater the sheet to about 50% consistency. Finally, this sheet is sent through a dryer which evaporates the remaining water until the desired final product moisture content is achieved. After drying, the sheet is fed to a cutter-layboy that cuts the sheet into .8m² sheets and stacks these cut sheets for baling and shipping to the end user by the finishing and shipping department. In the particular machine being studied, the sheet is slightly less than 6m wide.

Cross machine direction (CD) profile refers to the moisture and ream weight at points measured across the sheet. This is an important parameter for the machine operators as a bad profile, at the least will cause such problems as plugs and missed cuts in the cutter layboy area. Because the sheet shrinks as it dries, a poor moisture profile will also cause the sheet to twist in the dryer and the side of the sheet may rub against internal components of the dryer causing a sheetbreak. Unfortunately, the control of CD profile is complex and on the machine in question is a manual operation. Cross machine profile is critical for the paper industry and therefore CD control has been the subject of research over the years and remains a significant research area today.

Machine direction (MD) profile refers to the change in the average cross machine ream and moisture values with respect to time. The amount of water removed from any given component of the machine is dependent on the water content entering that component. If the sheet has too much water content when it enters the press section or dryer, then reliability problems will occur. These sections of course can be set up by operations to handle large deviations in moisture content by running slower rates, however, that impacts machine throughput. The goal of operations is to run the machine at as high a rate as practical while maintaining reliability. In order to do this, good moisture and weight control is required in the machine direction.

This control is effected by the headbox. The internal layout of a pulp machine headbox is illustrated in Figure 5.1. The amount of stock delivered to the wire is controlled by the stock consistency and by the exit speed of the jet leaving the headbox. There are several consistency control mechanisms associated with the headbox so consistency can be considered relatively constant for this study. Control of jet exit velocity is of interest here.

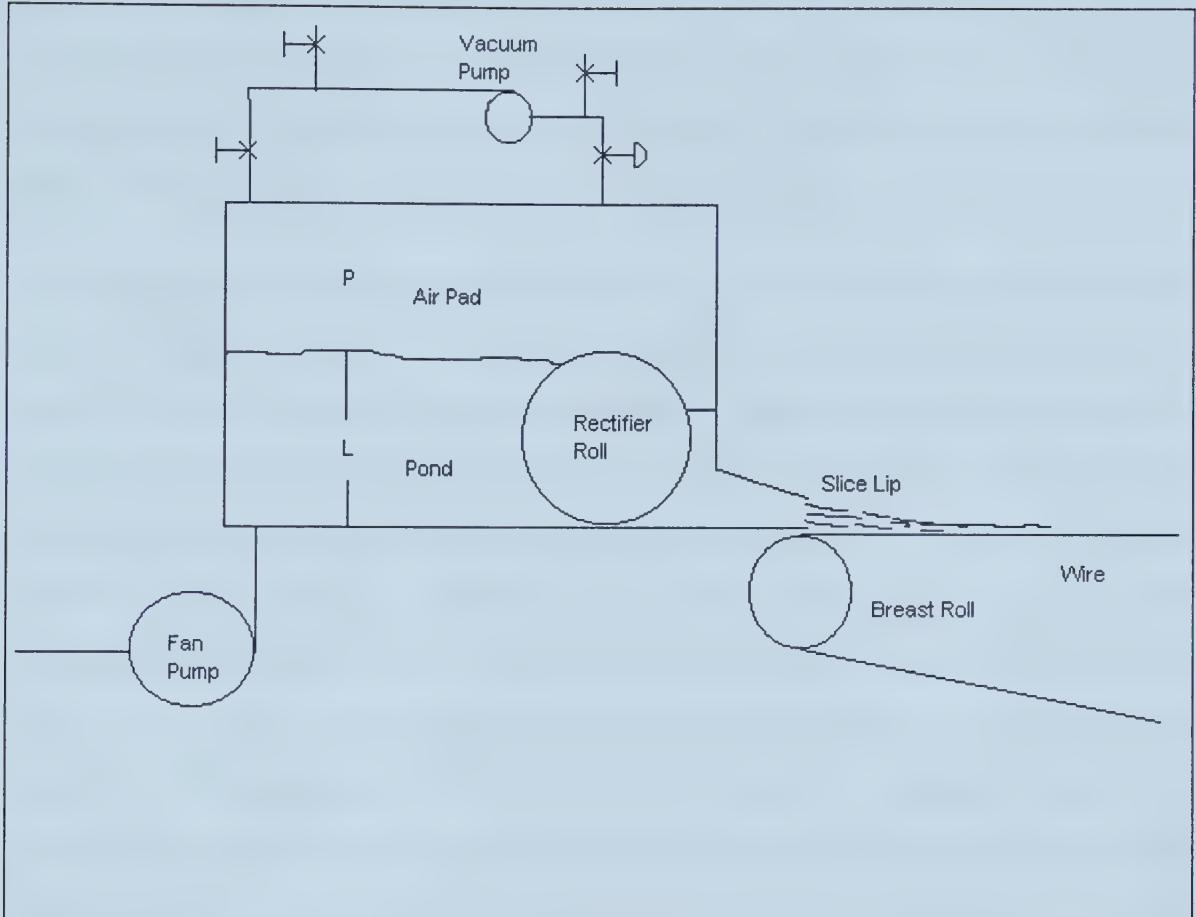


Figure 5.1: Headbox

Jet exit velocity is given by the equation

$$v = C_D \sqrt{2g(h_l + h_p)} \quad (5.1)$$

where

C_D is the coefficient of discharge for the slice opening

g is the gravitational constant

h_p is the head at the slice due to the air pad pressure in the headbox

h_l is the head at the slice due to the pulp level in the headbox (pond level)

C_D represents the efficiency of converting the potential energy due to the head into kinetic energy. Values less than 1 are caused by losses in the system including losses due to the geometry of the slice lip and losses due to the flow resistance imposed by the rectifier roll. C_D is calculated by simple mass balance equations and ranges from about .8 at slow speeds (~1.5 m/s) to .65 at high speeds (~3 m/s).

Jet exit velocity control is important from two perspectives. First, it can be seen that the mass of pulp leaving the headbox at any given time is proportional to the jet exit velocity. Therefore, changes in exit velocity will directly change the MD ream profile. However, the jet exit velocity also has an impact on the sheet formation and the dewatering characteristics of the sheet (drainage). One can imagine the fibres leaving the headbox as being in a random orientation with respect to the wire. If the jet exit velocity is greater than the wire speed, the ends of the fibres next to the wire are slowed to the wire speed while the other ends keep moving at jet velocity. Therefore, the fibres tend to be stretched out on the wire in the direction of the wire travel. If the jet exit velocity is less than the wire speed, then the fibres will also be stretched out in the direction of the wire. If however, the wire and jet exit velocities are nearly the same, then the fibres tend to settle on the wire in a random orientation. This orientation is not conducive to good drainage and a wetter sheet is produced which in turn may cause sheetbreaks. The ratio between jet exit and wire speeds is known as the jet to wire ratio or the rush/drag ratio. Studies and practical experience show that optimum formation occurs when the rush/drag ratio is .9 or 1.1. In other words, the machine operator aims for a 10% difference in speed between the jet and the wire. In practice, people generally aim for the jet to be faster than the wire so a rush/drag target of 1.1 is typical.

As pulp flows through the headbox, it tends to form small (~1-3 cm) flocks or denser regions of pulp. These flocks may be seen in ordinary writing paper by holding it up to the light and observing the structure. The goal of pulp (and paper) machine operators is to eliminate these flocks as they represent sources of difficulty in dewatering, and in the case of paper makers, a cost in terms of excess fibre being shipped. In the pulp machine headbox, the rectifier roll is one component aimed to reduce flock formation. The rectifier roll is a rotating hollow perforated roll positioned immediately upstream of the slice opening. This roll creates turbulence in the pulp flow as it moves through the perforations. For the rectifier roll to be effective, the level of the pulp in the headbox must be maintained at exactly the height of the rectifier roll.

If the level is too low, air enters the roll and is entrained into the slurry. Entrained air is extremely detrimental to drainage. If the level is too high, pulp will flow over the roll and directly into the slice. This not only results in a partially un-deflocculated pulp flow, but also changes the head at the slice. The largest problem occurs if the headbox level is varying between these two extremes over time.

The remaining manipulated variable to control total head then is the air pad pressure h_l . However, the pressure in the headbox and the level are tightly coupled. For a given pulp flow into the headbox, if the air pad pressure is increased, the jet exit velocity will increase causing a net loss of pulp from the headbox which will reduce the pond level. Likewise, if flow into the headbox is increased, then pond level will raise. Due to the relatively incompressible nature of the air pad, this rise in level will result in a significant increase in pressure. The headbox represents a very dynamic intercoupled process and good control requires coincident changes to both air pad pressure and pulp flows.

5.2 PULP MACHINE HEADBOX CONTROL

5.2.1 INTRODUCTION

Coincident changes to pulp flow and air pad pressure requires a good model of the headbox. However, this has been difficult in practice due to the non-linearity of the process over the operating range. This is seen not only in the change of the C_D as previously discussed, but also any non-linearities in the pulp delivery and air pressurisation systems.

There are four variables of interest in the control of the headbox.: the pond level and air pad pressure represent the target variables; the two manipulated variables are the fan pump and vacuum valve position. The fan pump is a large variable speed pump that delivers about 20,000-gpm pulp stock to the headbox. The speed of the fan pump is proportional to the flow of stock. The air pad is maintained by a vacuum pump which through a complex piping arrangement also pressurises the headbox. The air pad pressure (which can be less than atmospheric), is controlled by the amount the vacuum valve is opened.

The development of first principle relationships between the four variables is difficult due to losses in the system and lack of accurate instrumentation. However, (although insufficient sampling has been conducted to develop a model across the entire operating range), there are sufficient data collected to permit empirical modelling using traditional or non-parametric approaches,. Therefore modelling must be a semi-empirical approach combining the empirical model for one operating condition with a mechanistic model to extend the model to other unsampled operating points.

One of the uses of the model is incorporation into a control algorithm that provides proper fan pump and air pad pressure (vacuum valve) control as to effect the required jet velocity while maintaining pond level. This control can take one of several forms:

1. Independent PID control for each of the fan pump and vacuum valves.
2. Decoupled PID control.
3. Model Based Predictive Control.
4. Model Based Predictive Control used to determine PID parameters for decoupled PID control.

There is literature published on each of these approaches. Approach 1 is the typical control method, as it is the simplest. If operation is constrained to a narrow operating range and if changes to new operating speeds are not made too quickly, independent PID control provides satisfactory results. Pulp plant control personnel often feel that the added complexity of other approaches with their consequent increase in skill requirements of the controls group, out weigh the potential improved performance across a wider operating range. In other words, a reduction in effort can be made by the controls organisation by constraining the operating strategy.

Several researchers have attempted decoupling of the controllers. One example is given in Xia *et al* [1993]. The main deficiency with most of the decoupled algorithms is that the process is non-linear over its operating range. Therefore, a decoupling strategy that is successful at one point in the operating range will be less successful at another. Weyerhaeuser's Grande Prairie mill implemented a decoupled control strategy (not based on the Xia paper) but the implementation proved to be a failure and the control was removed. This failure was attributed to the fact that the decoupling model was accurate at only one

operating condition and as a result, the PID gains calculated from the model were ineffective for any other operating conditions. Therefore, although the strategy was successful for the operating range for which it was designed, any movement from that range resulted in a severe degradation in performance, to the point where the machine could not be operated.

Another common approach to headbox control is the use of model predictive control to develop the gains for a decoupled PID controller. The advantage to this approach is that the simplicity of PID control is maintained for the field instrumentation technicians while the advantages of predictive control are achieved. Naturally this approach only applies in certain circumstances, for example, when the time constants of the processes to be controlled are sufficiently slow. This approach is discussed in depth in Zurcher *et al* [1996].

Model Based Predictive Control has been successfully implemented by many of the control vendors specialising in Pulp and Paper control applications. These techniques are particularly suited to the high speed, high performance paper machine. Pulp machines with their slower time constants benefit less from the predictive control algorithm. However, as will be discussed in Section 5.3, there are some problems with the control of the pulp machine headbox that can be overcome with some form of model based control.

5.2.2 EMPIRICAL MODEL OF A PULP MACHINE

Previous research [Zurcher *et al.* 1996] has investigated the use of an empirical model based control strategy to develop PID tuning parameters. The empirical model of the pulp machine was based on data collected in 1991 from the Weyerhaeuser Canada (Grande Prairie Operations) pulp machine. The model (as developed in Zurcher *et al* [1996]) is

$$\begin{bmatrix} Y_p \\ Y_L \end{bmatrix} = \begin{bmatrix} \frac{2.34 - 2.34z^{-1}}{1 - .940z^{-1}} & \frac{-0.086}{1 - .942z^{-1}} \\ \frac{.116}{1 - .951z^{-1}} & \frac{.107}{1 - .940z^{-1}} \end{bmatrix} \begin{bmatrix} U_{fp} \\ U_v \end{bmatrix} \quad (5.2)$$

or for the pressure relationship

$$y_p(1-940z^{-1})(1-942z^{-1}) = (2.34 - 2.34z^{-1})U_{fp} - .086U_v \quad (5.3)$$

$$y_p = 1.882(1-471z^{-1})y_p z^{-1} + 2.34(1-z^{-1})U_{fp} - .086U_v$$

and for the level relationship

$$y_L(1-951z^{-1})(1-940z^{-1}) = .116(1-940z^{-1})U_{fp} + .107(1-951z^{-1})U_v \quad (5.4)$$

$$y_L = 1.891(1-473z^{-1})y_L z^{-1} + .116(1-940z^{-1})U_{fp} + .107(1-951z^{-1})U_v$$

where

Y_p is the output value for pressure,

Y_L is the output value for pond level

U_{FP} is the Fan Pump speed

U_v is the air pad pressure control valve

5.2.3 DISCUSSION OF BUMP TEST RESULTS

The modelled bump test results are shown in Figure 5.2 and 5.3. The original bump test data was destroyed after the model was built so these bump tests regenerate the bump test data from the model. The two bump tests chosen for inclusion here (out of four tests) are those of Fan Pump Speed to Pressure and Valve Position to Level. These two were chosen, as they are the normal control pairings in a headbox. The effect of the lead/lag pressure relationship can be clearly seen in the Fan Pump bump test.

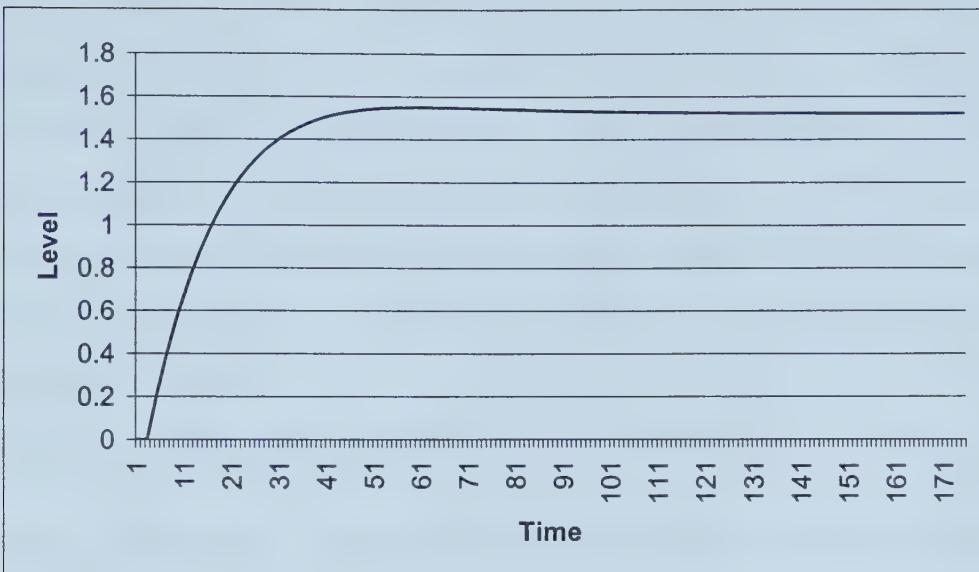


Figure 5.2 Valve Position versus Level Bump Test

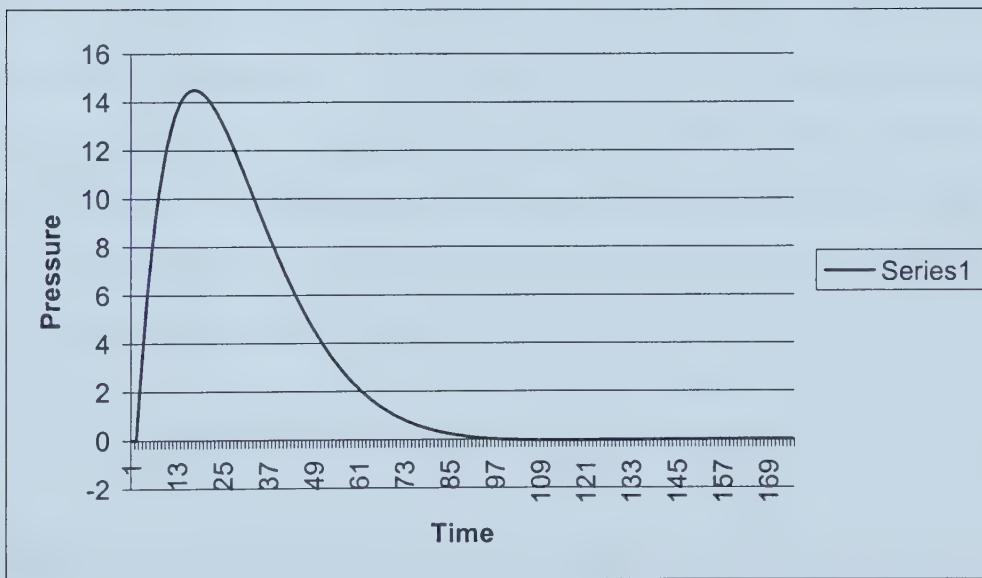


Figure 5.3 Fan Pump Speed versus Headbox Pressure

The fan pump bump test was a source of much discussion for the researchers as the lead lag relationship was unexpected by some. However the ideal gas law provided the required mechanistic explanation.

$$PV = nRT \quad (5.5)$$

As the fan pump increases speed, flow rate into the headbox increases rapidly. However, at the instant that fan pump speed increases, the total head in the headbox has not yet changed, and flow from the headbox remains constant. Therefore, inflow into the headbox is greater than outflow, and as a result, level increases and, the pressure in the airpad increases according to Equation 5.5. Level will continue to rise, and airpad pressure will continue to increase until such time as the total head at the slice causes outflow to equal to inflow. This response of air pad pressure to fan pump change is given by the numerator of the model, i.e., $Y_P = (2.34 - 2.34z^{-1})U_{FP}$.

At this point, the dynamics of the air system become predominant. The air system can be modelled as two components: a constant pressure air supply feeding the headbox through a control vale and an exhaust from the headbox through an orifice into a constant pressure sump. Therefore there is a constant movement of air through the headbox. As the pressure in the headbox increases due to the level increase, the pressure differential across the supply valve drops and as a result airflow into the headbox decreases. Similarly, the increased headbox pressure causes increased flow through the exhaust orifice. This mismatch of air flows will continue until the pressure in the headbox has reduced to the pre-bump level. The dynamics of this response is given by the denominator of the fan pump relationship, i.e., $Y_P = U_{FP} / (1 - .942z^{-1})$. The denominators of the full model are all approximately equal as they all describe the same phenomena, the flow of air through the inlet and outlet orifices of the headbox.

5.3 MODEL BASED CONTROL OF A HEADBOX

5.3.1 INTRODUCTION

The derivation of a model based predictive control law for the 2 by 2 pulp machine control system will follow the approach of:

- 1) developing a SISO predictive controller to illustrate the polynomial approach
- 2) converting the SISO polynomial example to a SISO matrix based format

- 3) develop a general nxn matrix based control law
- 4) apply the general nxn matrix form to a 2x2 control law as applied to the pulp machine headbox

The polynomial SISO derivation will be complete and shows the approach in detail. This allows the presentation of the matrix derivations to be streamlined. Due to their complexity, the general matrix based versions are more difficult to analyze if they are devolved into their constituent polynomials. However, in their matrix form, it is more difficult to follow the derivation. Therefore the complete derivation will be shown in step 1 and from this the matrix derivations can be easily followed.

5.3.2 GENERAL SISO PREDICTIVE CONTROLLER

The SISO derivation follows closely the derivation in Soeterboek [1992] and is quoted here, as that text is out of print and not easily found. Understanding the nomenclature used in the SISO case eases understanding of the MIMO controller.

Assume that the process being controlled follows the form

$$y(k+d) = \frac{B}{A} u(k-1) + \xi(k+d) \quad (5.6)$$

where

B is a polynomial

A is a monic polynomial

d is the deadtime

k is the current time

ξ is the disturbance.

For a prediction at time $k+d+i$ the process form is

$$y(k+d+i) = \frac{B}{A} u(k+i-1) + \xi(k+d+i) \quad (5.7)$$

It is assumed that the noise term ξ is given by

$$\xi(k+i) = \frac{T}{D} e(k+i) \quad (5.8)$$

Where T and D are polynomials

Using the Diophantine equation

$$\frac{T}{D} = E_i + q^{-i} \frac{F_i}{D} \quad (5.9)$$

$$\xi(k+i) = E_i e(k+i) + \frac{F_i}{D} e(k) \quad (5.10)$$

where $E_i e(k+i)$ represents future disturbances and $\frac{F_i}{D} e(k)$ is past disturbances.

Equation 5.7 can be rewritten as

$$y(k+d) = \frac{B}{A} u(k-1) + \frac{T}{D} (k+d) \quad (5.11)$$

Multiplying 5.11 by F/D and rearranging yields

$$\begin{aligned} \frac{F_i}{T} y(k+d) &= \frac{F_i B}{TA} u(k-1) + \frac{F_i}{D} e(k+d) \\ \frac{F_i}{D} e(k+d) &= \frac{F_i}{T} y(k+d) - \frac{F_i B}{TA} u(k-1) \\ &= \frac{F_i}{T} \left\{ y(k+d) - \frac{B}{A} u(k-1) \right\} \end{aligned} \quad (5.12)$$

From Equation 5.7 we have

$$\begin{aligned}
y(k+d+i) &= \frac{B}{A} u(k+i-1) + \xi(k+d+i) \\
&= \frac{B}{A} u(k+i-1) + E_i(k+d+i) + \frac{F_i}{D} e(k+d) \\
&= \frac{B}{A} u(k+i-1) + \frac{F_i}{T} \left\{ y(k+d) + \frac{B}{A} u(k-1) \right\} + E_i(k+d+i)
\end{aligned} \tag{5.13}$$

The five terms in Equation 5.13 can be described as

$y(k+d+i)$ Prediction of future y values

$\frac{B}{A} u(k+i-1)$ Past and future values of controller action u

$\frac{F_i}{T} y(k+d)$ Past values of y

$\frac{F_i}{T} \frac{B}{A} u(k-1)$ Past controller actions u

$E_i(k+d+i)$ Prediction of future noise

Taking the expectation of 5.13 and assuming that the noise is normal and zero mean yields

$$\hat{y}(k+\hat{d}+i) = \frac{\hat{B}}{\hat{A}} u(k+i-1) + \frac{F_i}{\hat{T}} \left\{ y(k+\hat{d}) + \frac{\hat{B}}{\hat{A}} u(k-1) \right\} \tag{5.14}$$

substituting the process model

$$\hat{y}(k+\hat{d}) = \frac{\hat{B}}{\hat{A}} u(k-1) \tag{5.15}$$

into Equation 5.14 yields

$$\hat{y}(k + \hat{d} + i) = \frac{\hat{B}}{\hat{A}} u(k + i - 1) + \frac{F_i}{\hat{T}} \{y(k + \hat{d}) + \hat{y}(k + \hat{d})\} \quad (5.16)$$

The choices for A,B,T, and D can follow certain forms depending on the type (GPC, DMC, etc.) of predictive controller being designed. Table 2.3 on page 46 of Soeterboek [1992] describes various forms for A,B,T,D for each type of controller. For a GPC controller with an ARIMAX process, A and B are the transfer function polynomials, T is an arbitrary polynomial and D=AΔ.

To separate past and future controller actions another Diophantine equation is used

$$\frac{\hat{B}}{\hat{A}} = G_i + q^{-i+\hat{d}} \frac{H_i}{\hat{A}} \quad i \geq \hat{d} + 1 \quad (5.17)$$

and when substituted into Equation 5.16 yields

$$\hat{y}(k + \hat{d} + i) = G_i u(k + i - 1) + \frac{H_i}{\hat{A}} u(k + \hat{d} - 1) + \frac{F_i}{\hat{T}} \{y(k + \hat{d}) + \hat{y}(k + \hat{d})\} \quad (5.18)$$

where

$G_i u(k + i - 1)$ is comprised solely of future controller actions and

$\frac{H_i}{\hat{A}} u(k + \hat{d} - 1)$ is comprised of past controller actions.

Rearranging the delay in 5.15 yields

$$\hat{y}(k + i) = G_i u(k + i - \hat{d} - 1) + \frac{H_i}{\hat{A}} u(k - 1) + \frac{F_i}{\hat{T}} \{y(k) + \hat{y}(k)\} \quad (5.19)$$

5.3.3 MATRIX DERIVATION OF A PREDICTIVE CONTROL LAW

Soeterboek [1992] develops a control law called the Unified Predictive Control Law which incorporates extensive setpoint and control tracking. For simplicity this feature has not been incorporated here.

Soeterboek's approach allows for each step of the prediction to have individual weights for the controller action and the deviation from setpoint. In most circumstances, the control engineer is satisfied with a single weight for all control steps. In other words, weighting between controller action and deviation from setpoint is required, but is constant over the control horizon. In this case, a single weight (ρ) on the control action suffices resulting in a streamlined control law which is derived in this section

Equation 5.19 can be written in matrix notation as

$$\bar{\hat{Y}} = \bar{\bar{G}}\bar{U} + \bar{\bar{H}}\bar{U} + \bar{\bar{F}}\bar{C} \quad (5.20)$$

where

$$\bar{\hat{Y}} = \left[\hat{y}(k+d+1), \dots, \hat{y}(k+H_p) \right]^T$$

$$\bar{U} = \left[u(k), \dots, u(k+H_p - \hat{d} - 1) \right]^T$$

$$\bar{C} = \left[\frac{1}{A} u(k-1), \frac{1}{A} u(k-2), \dots \right]^T$$

$$\bar{G} = \left[\frac{y(k) - \hat{y}(k)}{T}, \frac{y(k-1) - \hat{y}(k-1)}{T}, \dots \right]^T$$

$$\bar{\bar{G}} = \begin{bmatrix} g_0 & 0 & \cdots & 0 \\ g_1 & g_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{H_p - \hat{d} - 1} & \cdots & \cdots & g_0 \end{bmatrix}$$

$$\bar{\bar{H}} = \begin{bmatrix} h_{\hat{d}+1} & 0 & \cdots & 0 \\ h_{d+2} & h_{\hat{d}+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{H_p} & \cdots & \cdots & h_{\hat{d}+1} \end{bmatrix}$$

$$\bar{\bar{\bar{F}}} = \begin{bmatrix} f_{\hat{d}+1} & 0 & \cdots & 0 \\ f_{\hat{d}+2} & f_{\hat{d}+1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ f_{H_p} & \cdots & \cdots & f_{\hat{d}+1} \end{bmatrix}$$

H_p is the prediction horizon

To solve for the future controller actions the following cost function is utilised.

$$\mathbf{J} = (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}})^T (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}}) + \rho \bar{\bar{\bar{U}}}^T \bar{\bar{\bar{U}}} \quad (5.21)$$

$$\begin{aligned} \frac{\partial \mathbf{J}}{\partial \bar{\bar{\bar{U}}}} &= \frac{\partial}{\partial \bar{\bar{\bar{U}}}} \{ (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}})^T (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}}) + \rho \bar{\bar{\bar{U}}}^T \bar{\bar{\bar{U}}} \} \\ &= 2 \frac{\partial \bar{\bar{\bar{Y}}}}{\bar{\bar{\bar{U}}}} (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}}) + 2\rho \bar{\bar{\bar{U}}} \end{aligned} \quad (5.22)$$

where

$$\begin{aligned} \frac{\partial \bar{\bar{\bar{Y}}}}{\partial \bar{\bar{\bar{U}}}} &= \frac{\partial}{\partial \bar{\bar{\bar{U}}}} \{ \bar{\bar{\bar{G}}} \bar{\bar{\bar{U}}} + \bar{\bar{\bar{H}}} \bar{\bar{\bar{U}}} + \bar{\bar{\bar{F}}} \bar{\bar{\bar{C}}} \} \\ &= \bar{\bar{\bar{G}}} \end{aligned} \quad (5.23)$$

Therefore

$$\begin{aligned} \frac{\partial \mathbf{J}}{\partial \bar{\bar{\bar{U}}}} &= \frac{\partial}{\partial \bar{\bar{\bar{U}}}} \{ (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}})^T (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}}) + \rho \bar{\bar{\bar{U}}}^T \bar{\bar{\bar{U}}} \} \\ &= 2 \bar{\bar{\bar{G}}} (\bar{\bar{\bar{Y}}} - \bar{\bar{\bar{W}}}) + 2\rho \bar{\bar{\bar{U}}} \end{aligned} \quad (5.24)$$

Setting the derivative to zero and solving for \mathbf{U} yields

$$\begin{aligned}
& 2\bar{\bar{\mathbf{G}}}^T \left(\bar{\bar{\mathbf{GU}}} + \bar{\bar{\mathbf{HU}}} + \bar{\bar{\mathbf{FC}}} - \bar{\mathbf{W}} \right) + 2\rho \bar{\mathbf{U}} = 0 \\
& 2\bar{\bar{\mathbf{G}}}^T \bar{\bar{\mathbf{GU}}} + 2\bar{\bar{\mathbf{G}}}^T \left(\bar{\bar{\mathbf{HU}}} + \bar{\bar{\mathbf{FC}}} - \bar{\mathbf{W}} \right) + 2\rho \bar{\mathbf{U}} = 0 \\
& \bar{\bar{\mathbf{G}}}^T \bar{\bar{\mathbf{GU}}} + \rho \bar{\mathbf{U}} = -\bar{\bar{\mathbf{G}}}^T \left(\bar{\bar{\mathbf{HU}}} + \bar{\bar{\mathbf{FC}}} - \bar{\mathbf{W}} \right) \\
& \left(\bar{\bar{\mathbf{G}}}^T \bar{\bar{\mathbf{G}}} + \rho \right) \bar{\mathbf{U}} = -\bar{\bar{\mathbf{G}}}^T \left(\bar{\bar{\mathbf{HU}}} + \bar{\bar{\mathbf{FC}}} - \bar{\mathbf{W}} \right) \\
& \bar{\mathbf{U}} = \left(\bar{\bar{\mathbf{G}}}^T \bar{\bar{\mathbf{G}}} + \rho \right)^{-1} \bar{\bar{\mathbf{G}}}^T \left(\bar{\mathbf{W}} - \bar{\bar{\mathbf{HU}}} - \bar{\bar{\mathbf{FC}}} \right) \quad (5.25)
\end{aligned}$$

5.3.4 HEADBOX SISO CONTROLLER

As an illustrative example, a one step-ahead SISO deadbeat controller for the fan pump speed to headbox pressure will be developed. The polynomials for this example are derived from Equation 5.2. The control law is given by Equation 5.25. As this is a deadbeat controller, ρ is 0.

5.3.5 CALCULATION OF F, G, AND H

As T is arbitrary, it is chosen to be 1. D is chosen to be $A\Delta$ which is consistent with a GPC controller (Soeterboek [1992]). Using Equation 5.9

$$\begin{aligned}
\frac{1}{(1-0.94z^{-1})\Delta} &= E_i + q^{-i} \frac{F_i}{(1-0.94z^{-1})\Delta} \quad (5.26) \\
\frac{1}{(1-1.94z^{-1}+0.94z^{-2})} &= E_i + q^{-i} \frac{F_i}{(1-1.94z^{-1}+0.94z^{-2})}
\end{aligned}$$

Values for E and F are calculated to be

$$\begin{aligned}
E_1 &= [1] \\
F_1 &= [1.94 \quad -0.94] \quad (5.27)
\end{aligned}$$

G and H are calculated from equation 5.17

$$\frac{2.34 - 2.34z^{-1}}{(1 - 0.94z^{-1})} = G_i + q^{-i} \frac{H_i}{(1 - 0.94z^{-1})} \quad (5.28)$$

$$G_i = [2.34] \quad (5.29)$$

$$H_i = [-1.1404]$$

The control law as given in Equation 5.25 becomes

$$\bar{\mathbf{U}} = \left(\bar{\mathbf{G}}^T \bar{\mathbf{G}} + 0 \right)^{-1} \bar{\mathbf{G}}^T (\bar{\mathbf{W}} - \bar{\mathbf{H}} \bar{\mathbf{U}} - \bar{\mathbf{F}} \bar{\mathbf{C}})$$

$$[U_1] = ([2.34][2.34] + [0])^{-1} [2.34] \left([w_1] - [-1.1404][\bar{U}_0] - [1.94 \quad .94] \begin{bmatrix} y_0 - \hat{y}_0 \\ y_{-1} - \hat{y}_{-1} \end{bmatrix} \right)$$

$$= .427 \left([w_1] - [-1.1404][\bar{U}_0] - [1.94 \quad .94] \begin{bmatrix} y_0 - \hat{y}_0 \\ y_{-1} - \hat{y}_{-1} \end{bmatrix} \right) \quad (5.30)$$

The performance of this controller is seen in Figure 5.4 (Note that white noise has been added as a disturbance to the output).

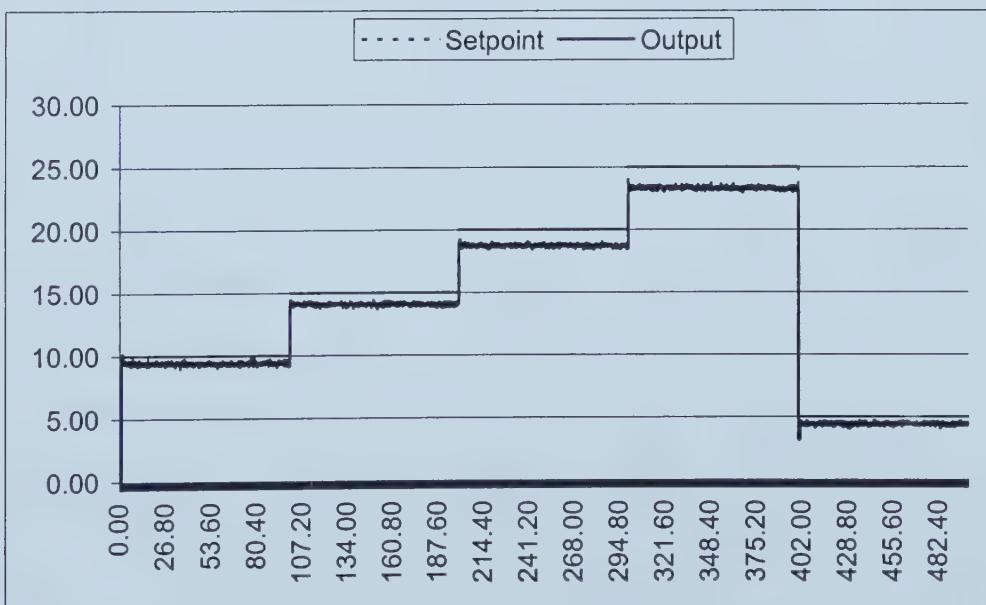


Figure 5.4: Deadbeat UPC Controller Response to Step Change

5.3.5 DEVELOPMENT OF A TWO STEP AHEAD CONTROLLER

A two step ahead controller can be developed using Equation 5.25. In this case the values for F,G,H are calculated as

$$\mathbf{F} = \begin{bmatrix} 1.94 & -0.94 \\ 2.82 & -1.82 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 2.34 & 0 \\ -1.404 & 2.34 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} -1.404 \\ -1.320 \end{bmatrix}$$

$$(\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T = \begin{bmatrix} .4274 & 0 \\ .0256 & .4212 \end{bmatrix}$$

Substituting these values into Equation 5.25 results in a controller that provides the same performance (as expected) as the polynomial derived version.

5.3.6 DEVELOPMENT OF A GENERAL MIMO CONTROL LAW

DEFINITIONS

In this section the following definitions will be used:

N_c = Number of Controllers

N_o = Number of Outputs

H_p = Prediction Horizon

H_c = Control Horizon

For general MIMO control using Soeterboek's [1992] Unified Predictive Control the following process model is used:

$$\underline{\underline{\mathbf{A}}}\hat{\underline{\mathbf{Y}}} = \underline{\underline{\mathbf{B}}}\Delta\underline{\mathbf{U}} + \underline{\underline{\mathbf{C}}}\underline{\mathbf{X}} \quad (5.31)$$

where

\underline{Y} are the past and predicted values of output \underline{Y} ($N_o \times H_p \times 1$)

\underline{U} are the past and future values of the controller ($N_c \times H_c \times 1$)

\underline{X} are the past and future noise terms ($N_o \times H_p \times 1$)

'A,B,C&D are matrices of factors in q^{-i}

For simplicity we assume that $D = A\Delta$

Using the Diophantine transformations described previously Equation 5.31 becomes

$$\hat{\underline{Y}}_f = \underline{F}\underline{Y}_p + \underline{G}\Delta\underline{U}_f + \underline{H}\Delta\underline{U}_p \quad (5.32)$$

where

$\hat{\underline{Y}}_f$ are the future (predicted) values of \underline{Y}

\underline{Y}_p are the past values of \underline{Y}

$\Delta\underline{U}_f$ are the future (calculated) values of \underline{U}

$\Delta\underline{U}_p$ are the past values of \underline{U}

(Note that for clarity in the MIMO example, the nomenclature has been changed from the previous sections to more clearly identify past and future values of the control and output vectors)

$$\hat{\underline{Y}}_f = \begin{vmatrix} \hat{\underline{Y}}_1 \\ \hat{\underline{Y}}_2 \\ \vdots \\ \hat{\underline{Y}}_n \end{vmatrix} \quad (5.33)$$

$$\hat{\underline{Y}}_i = \left| \hat{y}_i(k+d+1), \dots, \hat{y}_i(k+d+H_p) \right|^T \quad (5.34)$$

$$\underline{\underline{\mathbf{F}}} = \begin{vmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,1} & \cdots & \mathbf{F}_{1,N_C} \\ \mathbf{F}_{2,1} & \mathbf{F}_{2,2} & \ddots & \mathbf{F}_{2,N_C} \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{F}_{N_C,1} & \mathbf{F}_{N_C,2} & \cdots & \mathbf{F}_{N_C,N_C} \end{vmatrix}; \quad (5.35)$$

$$\underline{\underline{\mathbf{F}}}_i = \begin{vmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,p} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N_C,1} & f_{N_C,2} & \cdots & f_{N_C,p} \end{vmatrix}_i \quad (5.36)$$

where in the subscript of $f_{x,y}$ y is the degree of f and x is the i used to calculate the Diophantine equations. The degree of f (i.e. the value of p) is determined by the polynomial A.

$$\underline{\underline{\mathbf{Y}}}_p = \begin{vmatrix} \underline{\mathbf{Y}}_{p1} \\ \underline{\mathbf{Y}}_{p2} \\ \vdots \\ \underline{\mathbf{Y}}_{pn} \end{vmatrix}; \quad (5.37)$$

$$\underline{\underline{\mathbf{Y}}}_{pi} = \left| y_i(k+d), y_i(k+d-1), \dots, y_i(k+d-H_p-1) \right|^T \quad (5.38)$$

$$\underline{\underline{\mathbf{G}}} = \begin{vmatrix} \mathbf{G}_{1,1} & \mathbf{G}_{1,2} & \cdots & \mathbf{G}_{1,n} \\ \mathbf{G}_{2,1} & \mathbf{G}_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{G}_{n-1,n} \\ \mathbf{G}_{n,1} & \cdots & \mathbf{G}_{n,n-1} & \mathbf{G}_{n,n} \end{vmatrix} \quad (5.39)$$

where the subscript i,j refers to the effect of the jth controller on the ith output.

$$\underline{\underline{\mathbf{G}}}_{i,i} = \begin{vmatrix} g_{i,j,0} & 0 & 0 & \cdots & 0 \\ g_{i,j,1} & g_{i,j,0} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ g_{i,j,H_p-2} & \cdots & g_{i,j,1} & g_{i,j,0} & 0 \\ g_{i,j,H_p-1} & g_{i,j,H_p-2} & \cdots & g_{i,j,1} & g_{i,j,0} \end{vmatrix} \quad (5.40)$$

$$\underline{\mathbf{U}}_f = \begin{vmatrix} \underline{\mathbf{U}}_{f1} \\ \underline{\mathbf{U}}_{f2} \\ \vdots \\ \underline{\mathbf{U}}_{f2} \end{vmatrix} \quad (5.41)$$

$$\underline{\mathbf{U}}_f = \left| u_i(k), u_i(k+1), \dots, u_i(k+H_p - 1) \right|^T \quad (5.42)$$

where I refers to the controller number.

$$\underline{\mathbf{U}}_p = \begin{vmatrix} \underline{\mathbf{U}}_{p1} \\ \underline{\mathbf{U}}_{p2} \\ \vdots \\ \underline{\mathbf{U}}_{pn} \end{vmatrix} \quad (5.43)$$

$$\underline{\mathbf{U}}_{pi} = \left| u_1(k-i), u_2(k-i), \dots, u_{N_c}(k-i) \right|^T \quad (5.44)$$

$$\underline{\underline{\mathbf{H}}} = \begin{vmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \cdots & \mathbf{H}_{1N_c} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{H}_{Nc-1, Nc} \\ \mathbf{H}_{Nc1} & \cdots & \mathbf{H}_{Nc, Nc-1} & \mathbf{H}_{Nc, Nc} \end{vmatrix} \quad (5.45)$$

$$\underline{\underline{\mathbf{H}}}_{ij} = \begin{vmatrix} h_{i,j,1,k} \\ h_{i,j,2,k} \\ \vdots \\ h_{i,j,H_p,k} \end{vmatrix} \quad (5.46)$$

where k is the polynomial number (i from H_i in the polynomial example)

5.3.7 THE COST FUNCTION

As previously described, the extensive setpoint and controller tracking incorporated in UPC is not utilised here. This simplifies the derivation and application of the controller. The loss is the ability to provide weights to the individual steps of controller action and deviation from setpoint. In most cases, it is felt that the control horizon is small and the desired weight per step is constant. If the weights are constant, then controller and setpoint tracking can be incorporated into one constant, which, in this case; is the controller weighting. The MIMO control law however, does incorporate individual controller weights for each controller. The development of the streamlined MIMO controller closely follows Soeterboek's [1992] development of the SISO control law.

It is assumed that there will be a finite number (H_c) of predicted controller action steps and a finite prediction horizon (H_p) with H_c less than or equal to H_p . The vector of predicted controller actions \underline{U}_c is given by

$$\underline{U}_c = \left| u_1(k), \dots, u_1(k+H_c-1), \dots, u_n(k), \dots, u_n(k+H_c-1) \right|^T \quad (5.47)$$

The control action remains a constant after H_c steps.

\underline{U}_c maps to \underline{U}_f by M such that

$$\Delta \underline{U}_f = \underline{\underline{M}} \Delta \underline{U}_c \quad (5.48)$$

where

$$\underline{\underline{M}} = \begin{vmatrix} \mathbf{M}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{M}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{M}_{nc} \end{vmatrix} \quad (5.49)$$

$$\underline{\underline{M}}_i = \left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{array} \right] \left. \begin{array}{c} \\ \\ \\ \{ \\ H_c \\ \} \\ H_p - H_c \end{array} \right\} \quad (5.50)$$

Note that if the prediction horizon is the same as the control horizon, the M is the Identity matrix. The cost function is given by

$$\mathbf{J} = [\hat{\mathbf{Y}} - \underline{\mathbf{W}}]^T [\hat{\mathbf{Y}} - \underline{\mathbf{W}}] + \underline{\underline{P}} [\Delta \underline{\mathbf{U}}_f^T \Delta \underline{\mathbf{U}}_f] \quad (5.51)$$

where

$$\underline{\mathbf{W}} = \begin{vmatrix} \underline{\mathbf{W}}_1 \\ \underline{\mathbf{W}}_2 \\ \vdots \\ \underline{\mathbf{W}}_{no} \end{vmatrix} \quad (5.52)$$

$$\underline{\mathbf{W}}_i = \left| w_i(k+d+1), w_i(k+d+2), \dots, w_i(k+d+H_p) \right|^T \quad (5.53)$$

is the set point trajectory for each output, and

$$\underline{\underline{P}} = \begin{vmatrix} \underline{\underline{P}}_{1,1} & 0 & \cdots & 0 \\ 0 & \underline{\underline{P}}_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \underline{\underline{P}}_{Nc,Nc} \end{vmatrix} \quad (5.54)$$

$$\underline{\underline{P}}_{i,i} = \begin{bmatrix} \rho_{i,1} & 0 & \cdots & 0 \\ 0 & \rho_{i,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \rho_{i,H_p} \end{bmatrix}$$

is the matrix of weighting values for each (up to n_c) controller and ρ_{ij} is the weighting factor for controller i at step j. If all control steps are weighted equally then $\rho_{i,1}=\rho_{i,2}=\dots=\rho_{i,H_p}=\rho_i$.

Included in Equation 5.25 is the cost of the deviation from setpoint ($(\underline{Y}-\underline{W})^T(\underline{Y}-\underline{W})$) and the cost of the controller action ($P(\Delta \underline{U}_f^T \Delta \underline{U}_f)$). The parameter P allows a tradeoff between large controller actions and deviations from setpoint. When dealing with large control elements such as the fan pump, it is preferable to allow a little deviation from setpoint as opposed to continually changing the fan pump speed setpoint. The control law is obtained by minimising the cost function with respect to control action \underline{U}_c .

$$\frac{\partial \underline{\underline{J}}}{\partial \Delta \underline{U}_c} = 2 \frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_c} [\hat{\underline{Y}} - \underline{W}] + 2 \frac{\partial \Delta \underline{U}_f}{\partial \Delta \underline{U}_c} P \Delta \underline{U}_f = 0 \quad (5.56)$$

The partial derivatives $\frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_c}$ and $\frac{\partial \Delta \underline{U}_f}{\partial \Delta \underline{U}_c}$ must be determined.

$$\frac{\partial \Delta \underline{U}_f}{\partial \Delta \underline{U}_c} = \frac{\partial}{\partial \Delta \underline{U}_c} \underline{\underline{M}} \Delta \underline{U}_c = \underline{\underline{M}}^T \quad (5.57)$$

Solving $\frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_c}$ requires the use of the chain rule, i.e. $\frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_c} = \frac{\partial \Delta \underline{U}_f}{\partial \Delta \underline{U}_c} \frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_f}$

$$\begin{aligned} \frac{\partial \hat{\underline{Y}}}{\partial \Delta \underline{U}_c} &= \frac{\partial \Delta \underline{U}_f}{\partial \Delta \underline{U}_c} \frac{\partial}{\partial \Delta \underline{U}_f} [\underline{\underline{F}} \underline{Y} + \underline{\underline{G}} \Delta \underline{U}_f + \underline{\underline{H}} \Delta \underline{U}_p] \\ &= \underline{\underline{M}}^T \underline{\underline{G}}^T \end{aligned} \quad (5.58)$$

and

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \Delta \underline{\mathbf{U}}_c} &= 0 \\
&= 2 \underline{\underline{\mathbf{M}}}^T \underline{\underline{\mathbf{G}}}^T [\hat{\underline{\mathbf{Y}}} - \underline{\mathbf{W}}] \\
&\quad + 2 \underline{\underline{\mathbf{M}}}^T \underline{\underline{\mathbf{P}}} \underline{\underline{\mathbf{M}}} \Delta \underline{\mathbf{U}}_c \\
0 &= \underline{\underline{\mathbf{M}}}^T \underline{\underline{\mathbf{G}}}^T [\underline{\underline{\mathbf{F}}} \underline{\underline{\mathbf{Y}}} + \underline{\underline{\mathbf{G}}} \Delta \underline{\mathbf{U}}_f + \underline{\underline{\mathbf{H}}} \Delta \underline{\mathbf{U}}_p \\
&\quad - \underline{\mathbf{W}}] + \underline{\underline{\mathbf{M}}}^T \underline{\underline{\mathbf{P}}} \underline{\underline{\mathbf{M}}} \Delta \underline{\mathbf{U}}_c \\
\Delta \underline{\mathbf{U}}_c &= \left[\underline{\underline{\mathbf{M}}}^T (\underline{\underline{\mathbf{G}}}^T \underline{\underline{\mathbf{G}}} + \underline{\underline{\mathbf{P}}}) \underline{\underline{\mathbf{M}}} \right]^{-1} \\
&\quad * \underline{\underline{\mathbf{M}}}^T \underline{\underline{\mathbf{G}}}^T [\underline{\mathbf{W}} - \underline{\underline{\mathbf{F}}} \underline{\underline{\mathbf{Y}}} - \underline{\underline{\mathbf{H}}} \Delta \underline{\mathbf{U}}_p]
\end{aligned} \tag{5.59}$$

A more complete description of the derivation of these equations can be found in Zurcher *et al* [1996]. If the assumption is made that the control horizon is the same as the prediction horizon then equation 5.59 reduces to

$$\Delta \underline{\mathbf{U}}_c = \left[\underline{\underline{\mathbf{G}}}^T \underline{\underline{\mathbf{G}}} + \underline{\underline{\mathbf{P}}} \right]^{-1} \underline{\underline{\mathbf{G}}}^T [\underline{\mathbf{W}} - \underline{\underline{\mathbf{F}}} \underline{\underline{\mathbf{Y}}} - \underline{\underline{\mathbf{H}}} \Delta \underline{\mathbf{U}}_p] \tag{5.59}$$

5.3.8 DISCUSSION

Equations 5.59 and 5.60 are MIMO versions of Equation 2.121 in Soeterboek [1992]. Several simplifications are used. The first is the assumption that any control action past the horizon is constant. This is an intuitive simplification as in practice only the first control action will be utilised. After the first control action has been completed, the calculations are redone for the next iteration, so that new control actions will result. A second simplification is to calculate the change in control action. As a result fewer calculations are required than with the more complete Soeterboek approach. Even without the assumption about control and prediction horizons, there are many zeroes in the M matrix, which further simplifies the calculations, and the entire system can be implemented using Matlab®. Equation 5.60 will be used later in this chapter to determine controller actions for the headbox.

Relation	Description	A	B
1	Fan Pump to Pressure	$1-0.94z^{-1}$	$2.34-2.34z^{-1}$
2	Fan Pump to Level	$1-0.951z^{-1}$.116
3	Vacuum Valve to Pressure	$1-0.940z^{-1}$.107
4	Vacuum Valve to Level	$1-0.942z^{-1}$	-.086

Table 5.1. Bump Test Results

5.3.9 DEVELOPMENT OF A MIMO CONTROL LAW FOR A HIGHLY COUPLED HEADBOX

Using the data in Table 5.1 matrices for F,G,&H can be calculated as

$$F = \begin{vmatrix} 1.94 & -0.94 & 1.94 & -0.94 \\ 2.82 & -1.82 & 2.82 & -1.82 \\ 1.951 & -0.951 & 1.94 & -0.94 \\ 2.86 & -1.86 & 2.82 & -1.82 \end{vmatrix} \quad (5.61)$$

$$\mathbf{G} = \begin{vmatrix} 2.34 & 0 & -.086 & 0 \\ -.1404 & 2.34 & -.081 & -.086 \\ .116 & 0 & .107 & 0 \\ .1103 & .116 & .1006 & .107 \end{vmatrix} \quad (5.62)$$

$$\mathbf{H} = \begin{vmatrix} -.1404 & -.081 & 0 & 0 \\ -.132 & -.076 & 0 & 0 \\ 0 & 0 & .1103 & .1006 \\ 0 & 0 & .1049 & .0945 \end{vmatrix} \quad (5.63)$$

$$(\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T = \begin{vmatrix} .4110 & 0 & .3303 & 0 \\ -.0126 & .4110 & -.0096 & .3303 \\ -.4455 & 0 & 8.9778 & 0 \\ .0089 & -.4455 & -8.7802 & 8.9778 \end{vmatrix} \quad (5.64)$$

The results of this control strategy when subjected to a series of step changes in setpoint are shown in Figure 5.5

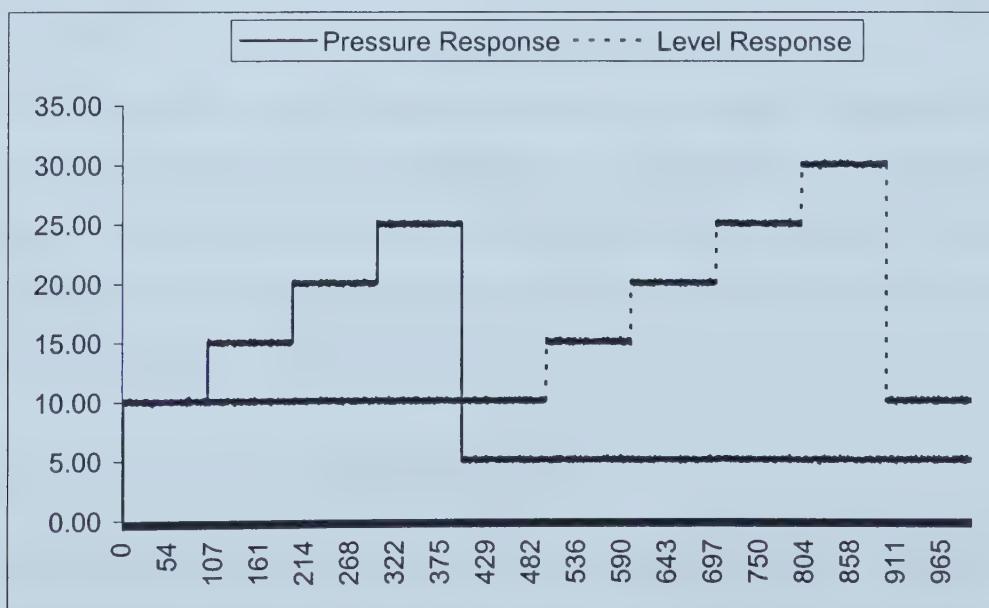


Figure 5.5: Process Output Using MBC MIMO Control

As can be seen, the decoupled model based control provides excellent decoupled control of pressure and level with no impact from coupled variables. In practice, this control scheme would of course need to be detuned as a result of model mismatch that would occur between the real process and the model. In this case, as the process is simulated, there is no model mismatch. Detuning can be accomplished one of the following methods:

constraining the control output

- ◆ by weighting the control action in the cost function (i.e. not deadbeat control as performed here), or
- ◆ by increasing the control horizon and weighting the control action

In the next section is an example of model mismatch between the control model and the actual process. In this case the control output is constrained and satisfactory control results despite the model mismatch.

5.4 PID AND APLM CONTROL OF A HEADBOX

5.4.1 INTRODUCTION

The purpose of this section is to show the application of APLMs in a real world control application. The control application is tested on a simulated headbox based on data taken from Weyerhaeuser's Grande Prairie pulp mill. In this example, the knowledge about the dynamics of the process are well known, which allows the accuracy of the modelling technique to be analysed. It was seen that the APLM was satisfactory and provided superior control when compared a well-tuned PID controller.

5.4.2 MODELLING THE HEADBOX USING APLMS

Modelling the headbox provides a significant challenge. To illustrate the APLM process a data set was generated using Equation 5.2. The APLM software was then used to generate a piecewise linear model of the generated data set. Ideally the APLM model would coincide with the equations used to generate the

data set. Despite several runs with various settings of s , an exact model never occurred. Further investigation was required to determine why.

Examining Equation 5.4, it can be seen that the auto-regression portion is very large. Expanding this equation yields

$$y_p = 1.882y_pz^{-1} - .894y_pz^{-2} + 2.34(1-z^{-1})U_{fp} - .086U_V \quad (5.65)$$

For the data-sets used to identify the model, a simple PID scheme was used to provide control values. This resulted in values for y_p that were very similar to each other. In this case $y_pz^{-1} \approx y_pz^{-2}$ and Equation 5.60 can be rewritten as

$$y_p = .988y_pz^{-1} + 2.34(1-z^{-1})U_{fp} - .086U_V \quad (5.66)$$

Therefore the current value for y_p can be predicted very closely by just using the previous value of y_p . In the steady-state data-sets examined, the contribution of the fan pump is less than 2% of the total pressure. In order to rectify this problem, a new data-set was generated that utilised a random value for U_{fp} that provided sufficient excitation for the system.. Care was taken when choosing excitation for the system that physically realisable controller actions were provided (even though this was a simulation exercise). As well, a conservative view was taken as to how much excitation would be provided so that the final data-set still had too little excitation for proper identification. However, the simulation was an accurate (if conservative) view of what would be done at the plant during a short machine outage if the machine had been available for testing.

The identification process using the APLM software provided a model of the system as

$$y_p = 1.915y_pz^{-1} - .919y_pz^{-2} + 1.827(1-z^{-1})U_{fp} - .153U_V \quad (5.67)$$

The major (autoregressive) components of the model have been correctly identified, however, the exogenous portions are incorrectly identified. However, the model is close enough for use in a control system if detuned properly.

5.4.3 PID CONTROL OF THE HEADBOX

Generally it is felt that the time constants in the headbox are slow enough that standard PID control is sufficient. However, in many mills, reliability has become of prime consideration and this control strategy has a direct impact on machine reliability. A machine outage, whether planned such as for a downtime, or unplanned such as for a sheet-break, is measured from the time the sheet is “taken off” the machine until the machine returns to full rate. Therefore machine reliability is impacted by the length of time that it takes to ramp the machine up to speed from the start-up speed of 300 fpm to operating speed of close to 600 fpm.

Figure 5.6 shows a simulation of the headbox pressure and level as the machine ramps up.

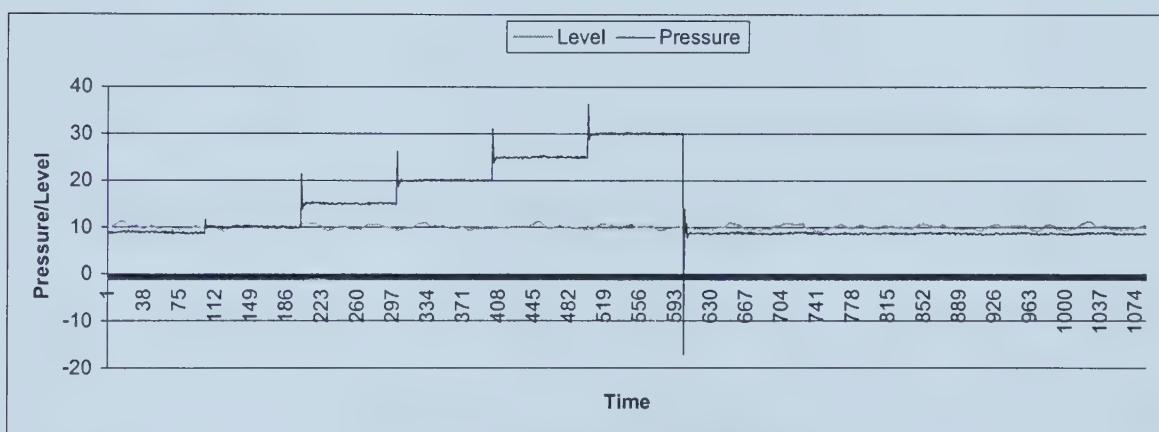


Figure 5.6 Simulated Machine ramp-up using PID Control

As can be seen, significant overshoot occurs at each speed-up of the machine. The last change, a drop from a pressure of 30 inches water to 8.67 simulates conditions when a sheet-break occurs and the machine is quickly slowed down to 300 fpm. Although the magnitude of the simulated negative overshoot never occurs in real life, an overshoot does occur and it is not unusual to have a wet end break during the ramp-down process.

In the simulation used to generate Figure 5.6, the PID loops were tuned to provide minimum variation during normal operation. As well, white noise to a level of .25 inches water was added. This corresponds to the field conditions and represents a signal to noise ratio of 35db.

In order to rectify the overshoot evident in this simulation, operations can utilise three tactics. First, speed ramp-up can be done slowly. However, this leads to a lengthening of the downtime due to the procedure of measuring downtimes from outage start to the time the machine is at full rate. Second, the loops can be run manually. Finally, the pressure loop can be de-tuned in order to minimise overshoot. In practice, operations has used all of these tactics from time to time. Figure 5.7 illustrates a start-up using a de-tuned PID controller

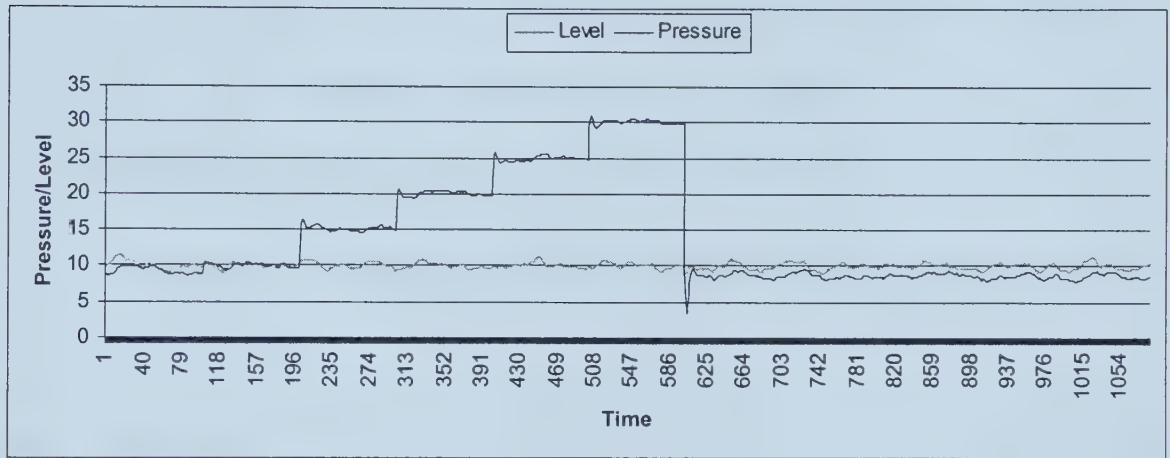


Figure 5.7 Simulated De-tuned Start-up

5.3.4 APLM CONTROL

A controller can be built for this process using the model (Equation 5.2) developed in Section 5.2.2. Because of the model mismatch and stability issues caused by the large autoregressive component, a minimum variance controller will not be successful. However, a two step-ahead controller is feasible. This simulation is shown in Figure 5.8

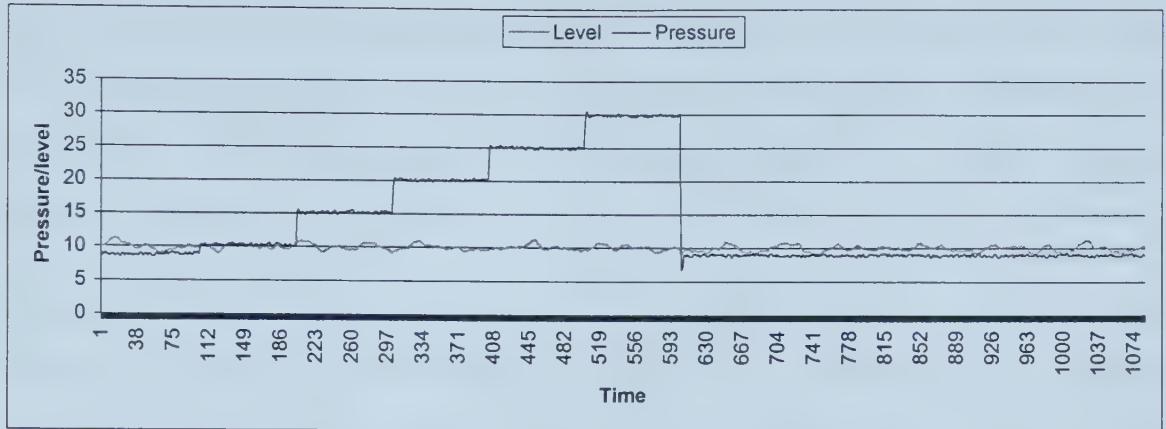


Figure 5.8 Simulated 2 Step Ahead Control

The controller for this simulation is the inversion of Equation 5.67 and is given as

$$U_{fp} = S(U_{fp}z^{-1} + (-y_{sp} + 1.915y_p z^{-1} - .919y_p z^{-2} - .153U_v) / -1.827) \quad (5.68)$$

where

y_{sp} = the set point

S = the step ahead factor

In this simulation the step ahead factor is set to .6 which means that the controller spreads the control action over two steps, 60% of the movement occurring in the first step and presumably 40% in the second step. Of course in reality, new control values are calculated at every step, so the second step is never utilised.

Control Scheme	Sum of Squared Error	Signal to Noise Ratio
Tuned PID	1475	23
De-tuned PID	832	25
Inverted Model Controller	117	33.7

Table 5.2 Comparison of Control Scheme results

Table 5.2 compares the error from set point for each control scheme. The sum of squared error measures the total deviation from the set point, while the signal to noise ratio column considers deviations from set point to be noise and then the signal to noise ratio can be used as a measure of control capability. This is an extension of a suggestion by Masters [1993].

The tuned PID scheme provides the worst control. However, this low rating is caused by the large deviations that occur when a set-point change occurs. If these points are eliminated from the calculation, then the Tuned PID provides comparable control to the inverted model controller. The de-tuned PID controller does not provide satisfactory control, although it does provide overshoot protection during set-point changes. Clearly, the best control is provided by the inverted model control, which despite model mismatch provides excellent matching to the set-point for all but the largest changes.

5.4 A NON-LINEAR HEADBOX MODEL

The previous section developed several control algorithms for a headbox assuming that its performance was nearly linear over the operating range. This is a not uncommon approach in industry, especially for machines that tend to run only one grade and rate. However, many machines do run multiple operating rates so this section summarises the use of APLM's to model a non-linear headbox.

The prime sources of non-linearity in the headbox are the square root term and the non-linear resistance to flow (indicated by C_D) in the velocity equation - Equation 5.1. A mechanistic model of the headbox was built which incorporated the results of the previous linear model, mechanistic knowledge, and anecdotal process data on the change of C_D with respect to jet exit velocity. Bump tests were performed on this simulation over the operating range (from 350 feet per minute to 650 feet per minute) with a 25 feet per minute step size. An APLM was used to identify the process and partition the operating range into quasi-linear segments. The relationships between pressure, level, fan pump control and valve position (Y_P , Y_L , U_{FP} , and U_V respectively) are

$$Y_P = A_1 Y_P z^{-1} + B_1 \Delta U_{FP} + C_1 U_V$$

$$Y_L = A_2 Y_L z^{-1} + B_2 U_{FP} + C_2 U_V$$
(5.69)

The results are shown in Table 5.3

Speed	A1	B1	C1	A2	B2	C2
350	0.854805	0.006761	0.07380433	0.967713	0.000708	-0.00702
375	0.854805	0.006761	0.07380433	0.967713	0.000772	-0.00702
400	0.854805	0.006761	0.07380433	0.967713	0.000937	-0.01699
425	0.869145	0.006761	0.06508141	0.879417	0.003824	-0.05871
450	0.869145	0.006761	0.06508141	0.879417	0.004273	-0.05871
475	0.869145	0.016642	0.06508141	0.879417	0.004808	-0.05871
500	0.869145	0.016642	0.06508141	0.879417	0.005272	-0.05871
525	0.90162	0.016642	0.04906827	0.914291	0.004278	-0.04358
550	0.90162	0.024073	0.04906827	0.914291	0.0048	-0.04358
575	0.90162	0.024073	0.04906827	0.914291	0.005331	-0.04358
600	0.93282	0.028125	0.03343004	0.932432	0.004619	-0.03358
625	0.93282	0.03168	0.03343004	0.932432	0.005107	-0.03358
650	0.93282	0.034615	0.03343004	0.932432	0.005683	-0.03358

Table 5.3 Identification of a Non-linear Headbox

Table 5.3 shows that generally at slow speeds a single model is sufficient. However, as the speed increases, the range over which a single model is valid diminishes. All of the values obtained are consistent with process knowledge except for B_2 the level to fan pump speed relationship. This value would be expected to be consistently increasing over the range. The fact that it doesn't is attributed to its small contribution to overall level, therefore, there can be considerable error in the identification process.

with little impact on the quality of the result. Other than this minor issue, the APLM identification matches well with the process knowledge contained in the simulation.

This illustrates the power of the APLM to partition the problem space into quasi-linear ranges. The control engineer has several options available to them at this point then depending on the their control environment. Several gain-scheduled PID controllers could be built and tuned using the above data which would provide fair control of the headbox. Likewise a scheduled model based controller could be designed, however, as the previous section illustrated, excellent results will be obtained by simply inverting the APLM models given above.

This section therefore illustrates the power of the APLM in breaking down control problems into linear partitions. This allows sites with minimal control expertise to implement simple linear control algorithms while achieving some of the benefits of non-linear control.

5.5 SUMMARY

In this chapter the use of model based control schemes was investigated. First a generic MIMO control law was developed using the matrix based differentiation techniques described in the previous chapter. An illustrative example of Model Based Control was developed for a headbox simulation. This simulation was based on bump test data taken from Weyerhaeuser's Grande Prairie pulp mill in 1991. As the simulation for the process was based on detailed bump test data it is felt to represent an accurate view of the pulp machine operation.

A MIMO Model Based Control scheme was then developed using an inverted APLM. The APLM was trained using the simulated bump test data. The training and data gathering was performed using very conservative approaches that might be found in the field. The APLM was able to develop a linear model of the process, however, there was some model mismatch due to the highly autoregressive nature of this process. From this model, a model predictive control scheme was developed. This is in contrast to current thinking which says that processes with large time constants do not benefit from model based control.

Several PID controllers were designed which showed that a well-tuned controller would generate excessive overshoot when a rate change occurred. This phenomena is seen in the real process and necessitates taking the process off control when rate changes are required. As well, these same dynamics limit the speed with which the machine can be ramped up after a shutdown. These PID based control schemes were compared to the APLM generated control scheme. It was found that the APLM based control scheme was equal to the best PID schemes during normal operation and superior when rate change effects were considered.

Importantly, it was shown that APLM developed models could work well when applied to difficult dynamic processes. The headbox has particular characteristics that make it very hard to identify. It was seen that the autoregressive portion of the process represented 99% of the current value. With this much autoregression, there was insufficient exogenous excitement to properly identify the model. As a result the APLM model did not match the actual process as well as might be desired; however, the match was sufficient for superior control results without resorting to excessive filtering or data manipulation.

Because the original bump test data was taken at only one rate, the machine was not modelled at other operating rates; however, it is known that the headbox represents a non-linear process as it is essentially a large tank with a hole in the bottom. As stated in the previous chapter, one of the strengths of APLMs is their ability to identify multiple models. Further work in this area will be to collect more process data that represents the entire operating range. This will allow a multi-segment APLM to be generated which can then be used to generate a multi-segment model based control scheme. As discussed previously, a variable is required to determine when to change segments in the control scheme. In this case, the fan pump speed can be used as an indicator of when to change segments. Once the training is complete, the APLM can be put into operation.

CHAPTER 6

6.0 PULP STRENGTH ANALYSIS

6.1 OVERVIEW

Pulp strength is the ability of a pulp sheet to resist breakage and the test procedure is similar in concept to tensile testing for metals. This parameter is important to paper makers and consumers as it is one of a few variables that indicates how well the sheet will resist breakage which can occur either in the papermakers' process, or in the consumers' use of the product. A more extensive description of the pulp strength issue may be found in Zurcher[1995] or Seborg *et al* [1997].

The research into pulp strength from a fibre morphology perspective is extensive (see for example Seborg *et al* [1997]). However, there is little research from the macro perspective of how a pulp plant can impact pulp strength and what manipulated variables should be utilised. A neural network with 43 variables has been successfully developed to model pulp strength from this macro perspective (Zurcher[1995]); however, due to the use of a non-MNNN approach, no knowledge could be obtained by direct examination of the network. A sensitivity analysis was conducted, but this did not cover the complete operating range and was not developed in accordance with the knowledge development methodology outlined later in this Chapter. The main objective of this part of the research was to rectify that outage by replacing the neural approach used previously with a new modelling technology that will allow knowledge development.

The purpose of this chapter is to analyse the key indicators of pulp strength from the geometric perspective described in Section 3.1.2. Of prime interest will be the degree of non-linearity present in this problem. Several attempts have been made to study this problem using traditional linear methods. The results have not been published and details are not available, but in general, the results show varying degrees of success with no analysis being completely acceptable. The question that must be answered is the degree of non-linearity present in this problem and whether this in conjunction with the linear analyses is contributing to the poor results. To answer this question an APLM will be built of the problem and the key impactors on pulp strength will be determined. These variables will then be examined over the range of interest to

identify the degree of non-linearity as indicated by the number of identified segments, their size, and the angle between adjacent segments.

Also included in this chapter is a description of the dataset used, and the methodology for building an APLM model follows a brief review of what pulp strength is. In the course of building this model, a neural network was built to help analyse what data should be included in the dataset. The results from this neural network approach will be discussed and contrasted to the findings from the APLM. As well, a brief discussion on the dangers of sensitivity analysis as performed by many neural network software packages is undertaken and it will be shown how APLMs avoid this problem.

6.2 INTRODUCTION

A tree is made up of roughly equal parts of cellulose fibre, lignin, and water. Cellulose is the desired end product of a pulp plant and the pulping process is responsible for separating the cellulose fibres from the lignin. In a Kraft mill the pulping process consists of "cooking" wood chips in a caustic liquor of NaOH at high temperature and pressure. The pulping process, and the bleaching process that follows it, has a detrimental impact on the cellulose fibres resulting in a loss of pulp strength. Operating personnel have identified approximately 125 variables in the pulp plant that can indicate changes in pulp strength. Some of these variables are manipulated variables while the majority are measured variables. Because of the complexity of the processes involved, only generalisations can be made about the relationship between any given variable and the finished product pulp strength.

Market Northern Softwood Kraft (NSK) pulp plants such as Weyerhaeuser are often concerned about pulp strength. Market pulp is not used directly by the end consumer rather the primary customers of market NSK pulp are papermakers who produce a variety of products ranging from fine paper (writing, cigarette, etc.); to tissues and towels for the consumer market. In each case, the market NSK pulp is mixed with other types of pulp to produce a final product that meets strength, softness, and cost parameters. NSK pulp is of a higher cost than other pulps, however it provides strength. The word Kraft in NSK is German for strong. For example, in making writing paper, a papermaker would use significant recycled paper, hardwood, Southern Softwood and other furnish. However, without the strength provided by the NSK, the

papermaker would experience significant reliability problems as the paper sheet produced would not be strong enough to run through the papermaking and coating process without breaking. Therefore, from a papermaking standpoint it is important to use NSK pulp in order to keep production costs down. However, from a cost standpoint, the NSK pulp is typically the most expensive furnish so the papermaker will use as little as possible. As well, the stronger NSK pulp also has different printing and softness qualities than the softer cheaper furnishes. For example, toilet tissue made with too much NSK will be hard and paper like. From a papermaker perspective then, it is important not only to keep the amount of NSK furnish at a minimum, it also must be consistent with the recipe being used by the papermaker.

The discussion so far has focused on the amount of NSK furnish used. However, the strength of the furnish has an almost equal impact to the amount. In other words, if NSK pulp A is 10% stronger than NSK pulp B, the papermaker would have to use about 10% less of pulp A to get the same results achieved with pulp B. If the papermaker were using pulp B and then switched to pulp A without reducing the usage, the paper produced would have different qualities than that produced previously. Despite the obvious importance of pulp strength to the papermaker, it is unusual for the papermaker to actually test the pulps used for strength. The papermaker relies on pulp from a given supplier to be consistent over time. Because of the way pulp plants are run, this is usually the case. However, there are seasonal variations in strength that occur. As well, at the Grande Prairie plant, the strength of the pulp has gradually increased over the last 5 years to the point where the excess strength is causing customer concerns.

Therefore the Grande Prairie plant is keenly interested in understanding the causes of these strength changes and to learn what variables in the plant can be used to provide improved control. Control in this context is over a long period of time. From a total plant perspective the time constant of this process is measured in days. Also because of the nature of the problem, a static model is sufficient. With increased data it may be that a dynamic model becomes of interest later on, but for the present, just developing process knowledge is sufficient.

6.3 INITIAL APLM AND NN MODELS OF PULP STRENGTH

6.3.1 INITIAL APLM

Initially an APLM was built using all of the 125 input variables identified by the plant as having possible impacts on pulp strength. It was found that the model would not converge. Despite the fact that there were over 1000 data points, the reason for the non-convergence was determined to be a lack of data.

In m -dimensional space, it requires m points to define a plane. For example, in \mathbb{R}^3 , 3 points define a plane. In the pulp strength study, there are 125 inputs and 1 output, so this problem is in \mathbb{R}^{126} , and 126 points are required to define a hyper plane. In the year being studied there were 1074 valid pulp strength samples. At 126 points per segment this represents a maximum of 8 segments in the dataset without any regression. If there were no noise in the system then it would be feasible to force the software (by choosing an extremely large value of $|s|$) to generate an APLM with 8 segments. However, if there were any noise, then a relaxed fit would be desired to allow some averaging which would provide some filtering of the noise. If there is some curvature to the system, then at least two segments will be required which means that at best each segment would be the average of 4 segments which clearly provides too little opportunity for adequate filtering.

If an extremely large value of $|s|$ is used so that all 8 segments are identified, the problem any noise would contribute can be easily calculated. The 8 segments would be identified at random from all possibilities in the dataset. Determining the number of possibilities is a simple calculation

$$P = \sum_{i=1}^k i$$

k = # of datapts - dimension size - 1

(6.1)

Therefore, in the tensile problem there are 578,223 possible segments of which 8 would be chosen. Clearly, any noise in the system would cause the slopes of the 8 segments to be highly random and provide little knowledge about the underlying relationships.

Through this analysis it can be seen how significant the data problem is for the modeller. This analysis however, is possible for APLMs and provides exact answers for the APLM modeller. The APLM situation can be contrasted to the neural network situation.

The minimum number of exemplars to use in a neural network dataset has no fixed answer. Masters [1993] provides a “rule of thumb” that the number of exemplars should be at least twice the number of weights and preferably four times. Other authors have different rules of thumb. There are two problems with these “rules of thumb”. First is that the actual number of exemplars required is dependent on the information contained in the dataset (a point which many authors overlook). Unfortunately, this leads to a “Catch 22” as there is no way to determine what information the dataset contains without the analysis. Second, modern neural tools, in an attempt to ease the development burden, automatically generate the neural network and do not provide information about the number of nodes nor the connection pattern. In this case, even where the dataset topography is known, the modeller can not translate that into a required number of exemplars.

In this section however, it has been shown how a user of APLMs can directly calculate how many exemplars are required for a given linear model. This does not guarantee success, but can indicate if the APLM is doomed to failure due to lack of data. As will be seen in a later section, the APLM does provide statistics on how many points were included in each linear segment. If in a 2-segment, 10-D model with 1000 exemplars, the APLM indicates that 990 exemplars were used in one segment and 10 in the other, the modeller can see that there is a problem. The neural modeller has no data available to them to indicate this type of problem. Therefore, the APLM modeller has much more data available about the internal workings of the APLMs and how its results were generated.

6.3.2 NEURAL NETWORK ANALYSIS

Through a separate research effort, a neural network had been built on a partially reconstructed dataset that spanned the time period contained in the APLM data set used here. The year’s data that comprised the APLM data contained over 9600 data-points. However, most exemplars had no pulp strength readings. Elimination of those samples that had no pulp strength data or had other errors reduced the dataset to the 1074 exemplar size. The dataset used in the neural network study is termed partially reconstructed as the

missing pulp strength data was interpolated using a boxcar scheme. This was sufficient for the purposes of the research project. A commercial neural network package provided by Pavilion was used to perform this other research. Figure 6.1 shows the neural network's predicted values versus actual pulp strength values. Figure 6.2 shows a scatter plot of predicted versus actual values. Table 6.1 tabulates the results of a sensitivity analysis by the neural network software. The sensitivity analysis is conducted by setting all inputs to their average values and then varying one input at a time by $\pm 5\%$. Sensitivity is then measured as percent change in output. Although the sensitivity is only conducted at the average operating range, it was felt that the analysis was sufficient to indicate the variables that had the biggest impact.

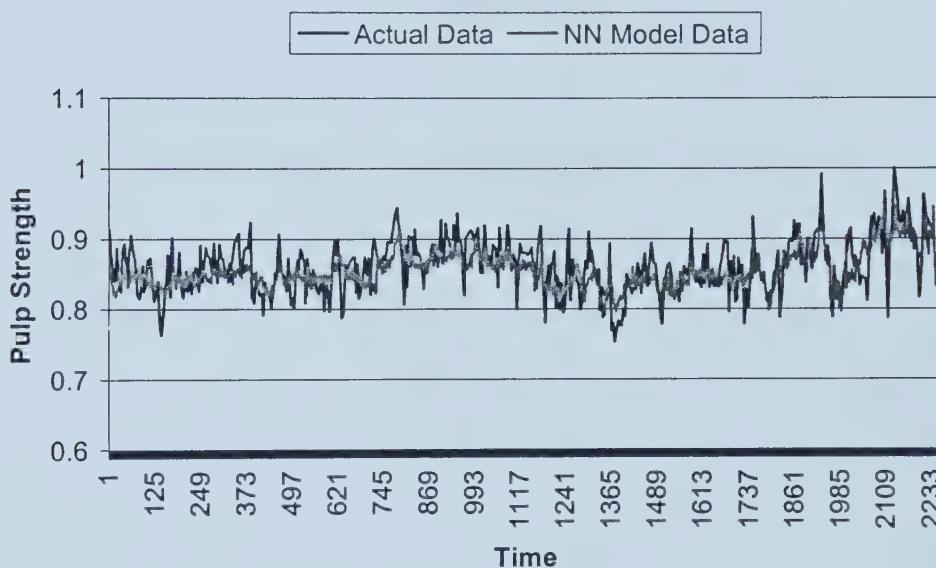


Figure 6.1 Predicted versus Actual Pulp Strength

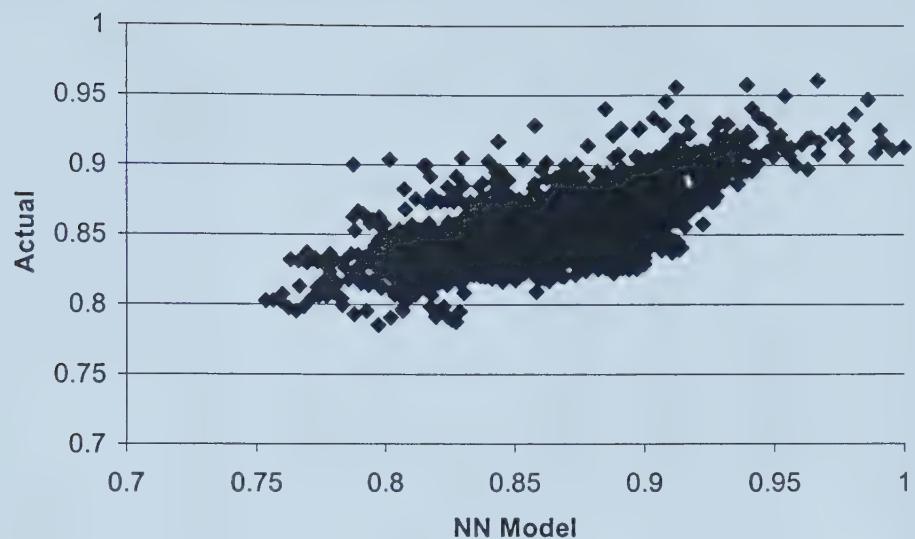


Figure 6.2 Predicted versus Actual Pulp Strength Scatter Plot

Rank	Name	Weight	Rank	Name	Weight	Rank	Name	Weight
1	V	0.268	35	Ream	0.049	69	720	0.028
2	2012	0.229	36	448	0.049	70	462	0.028
3	2272	0.157	37	252	0.048	71	189	0.027
4	454	0.145	38	89	0.048	72	318	0.027
5	2137	0.135	39	383	0.047	73	1101	0.026
6	03FC062.PV	0.114	40	847	0.047	74	272	0.025
7	03FY062.PV	0.113	41	921	0.045	75	399	0.023
8	M	0.108	42	271	0.045	76	944	0.022
9	416	0.103	43	1500	0.045	77	192	0.022
10	1412	0.096	44	81	0.043	78	23	0.021
11	OFC122.PV	0.093	45	03FI062A.PV	0.04	79	247	0.021
12	1559	0.091	46	135	0.04	80	2177	0.021
13	305	0.088	47	201	0.039	81	2246	0.021
14	137	0.087	48	845	0.038	82	133	0.021
15	923	0.086	49	1624	0.038	83	53	0.019
16	191	0.083	50	300	0.038	84	251	0.019
17	246	0.083	51	138	0.037	85	437	0.018
18	59	0.082	52	438	0.037	86	1804	0.016
19	299	0.081	53	249	0.036	87	1277	0.016
20	1508	0.08	54	1805	0.036	88	1470	0.016
21	1558	0.071	55	03FC063.PV	0.036	89	47	0.015
22	2109	0.071	56	248	0.035	90	M	0.014
23	536	0.068	57	90	0.035	91	25	0.013
24	349	0.064	58	422	0.033	92	1638	0.012
25	03FI062B.PV	0.063	59	1409	0.031	93	1223	0.012
26	1869	0.059	60	1552	0.031	94	203	0.01
27	297	0.058	61	234	0.03	95	1632	0.009
28	2345	0.057	62	1222	0.03	96	1356	0.009
29	413	0.054	63	942	0.029	97	940	0.008
30	719	0.054	64	1066	0.029	98	945	0.006
31	86	0.053	65	196	0.029	99	303	0.006
32	1410	0.053	66	309	0.029	100	943	0.006
33	289	0.05	67	306	0.028			
34	266	0.05	68	88	0.028			

Table 6.1 Sensitivity Analysis Results

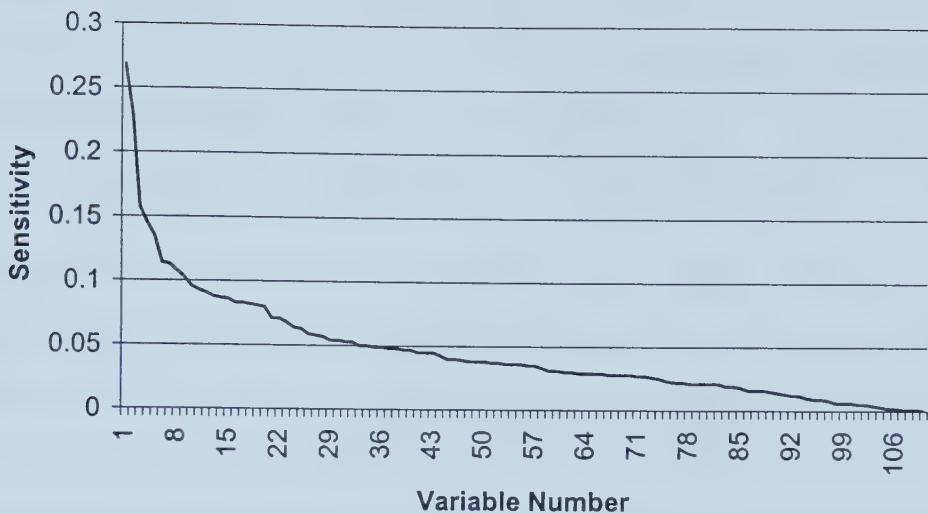


Figure 6.3 Sensitivity Results

6.3.3 ISSUES WITH NEURAL NETWORK MODEL

Several issues with the neural network approach arise due to poor dataset formulation and concerns regarding the sensitivity analysis approach. The data set utilised in this test comprises a significant number of variables. In Zurcher [1995] a much smaller dataset (in terms of the number of variables) was utilised. As the neural network model (or any other modelling technology) is attempting to extract some form of knowledge from the data, it is important to ensure the knowledge is presented to the modelling tool in as robust a manner as possible. When techniques that handle correlated data are used, it is especially important to limit the number of correlated variables as much as possible. Variable selection must be made based on the concept of the knowledge being contained, as opposed to a helter-skelter presentation of data. For example, assume that a neural network models the function $z=f(x,y)$. If x and y are related such that $y=f(x)$, then although the neural network may accurately predict the output z from the inputs x and y , there is no guarantee where it will get its knowledge to do so. It may determine that $z=f(x)$ or $z=f(y)$, or any combination of $z=f(x,y)$. For example, if $z=2x+y$ and $y=2x$, when a sensitivity analysis is conducted, it may be determined that the neural network “learned” $z=4x$, $z=2y$, $z=3x+.5y$ or any other valid combination of x and y . Therefore, if possible, it is important to eliminate all known highly correlated variables from a

dataset, in order to limit the sources of knowledge for the neural network to deal with. In examining the dataset, it was found that this was not done, and several variables contained duplicate knowledge. Unfortunately due to other issues, this problem could not be corrected prior to its inclusion in the neural network model. Therefore the results of the neural network sensitivity analysis are suspect.

A review of the neural network sensitivity analysis output seems to show that the analysis is flawed. Variables that have consistently placed in the top ten are much lower in value in this analysis. Of particular note is cooking rate, which is an indication of how the pulp is cooked in the digester. This typically is rated highly and makes sense from a process knowledge standpoint. This sensitivity analysis however, did not rate as significant those variables that indicate cooking rate. Therefore, the sensitivity analysis conducted by the neural network was considered to be suspect, although some of the variables that it rated highly do correspond with other sources such as previous studies and process knowledge.

In addition to concerns with the dataset, there is a concern with the manner in which sensitivity analyses are conducted by most neural network software. The concept of measuring output changes due to changes in inputs that are held near their average value, does not reflect the non-linear nature of most of the process that are being analysed by neural network software. Even given a perfect dataset so the previously mentioned correlated variable problems do not occur, it is unreasonable to consider that a non-linear process will exhibit the same sensitivity around the midpoint as it does in other areas of its operational space. This is one of the problems that is being rectified by the APLM technology. The presentation of multiple linear hyperplanes allows the modeller to see the impact across the entire function space, and not just at the midpoint.

Despite the issues with the dataset and the neural network approach, it was decided that the sensitivity analysis was a good starting point for this particular APLM analysis.

6.4 APLM MODEL OF PULP STRENGTH

As can be seen from Figure 6.3, the neural network indicated that only the first several variables showed any relation to pulp strength. Using this knowledge it was decided to build an APLM that utilised the top

10 variables as indicated by the neural network sensitivity analysis. In many cases, the choice of variables by the neural network was counter intuitive based on process knowledge. In other words, process knowledge would not tend to support these choices as being prime candidates for a pulp strength model. However, it was decided to use these variables as some were consistent with process knowledge and it was felt to be a good test of the APLM software.

Graphs of the prediction model are shown in Figure 6.4 and the scatter plot of modelled versus actual strength data is illustrated in Figure 6.5. The model was composed of 8 linear segments, of which 5 segments accounted for less than 10% of the samples. Three segments modelled 90+% of the dataset.

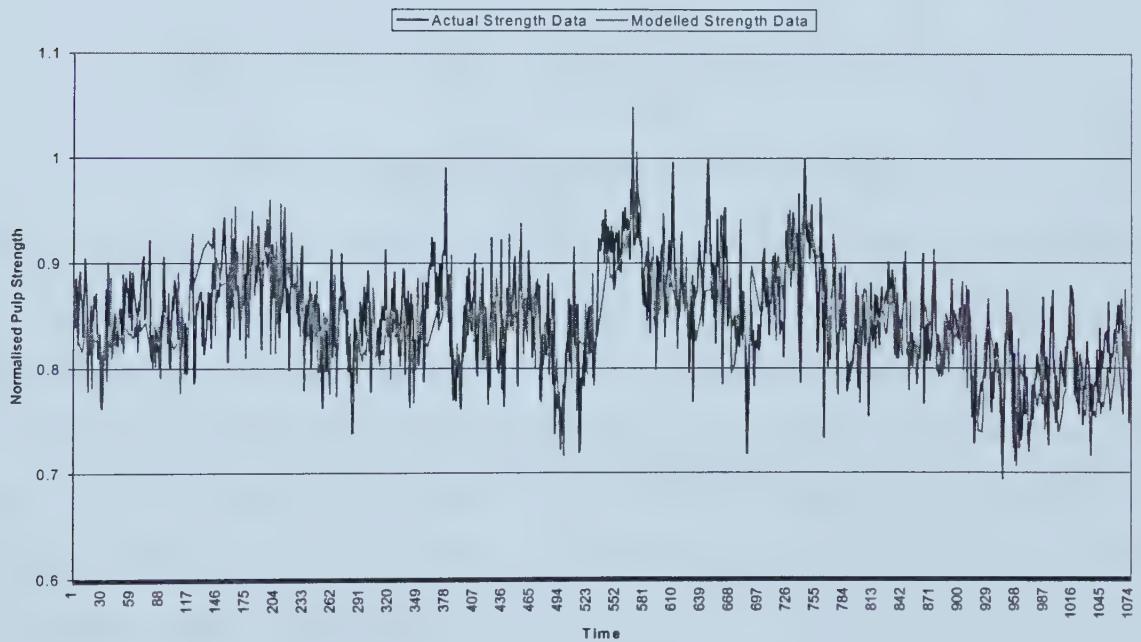


Figure 6.4 Trend Plot of Modelled versus Actual Strength Data

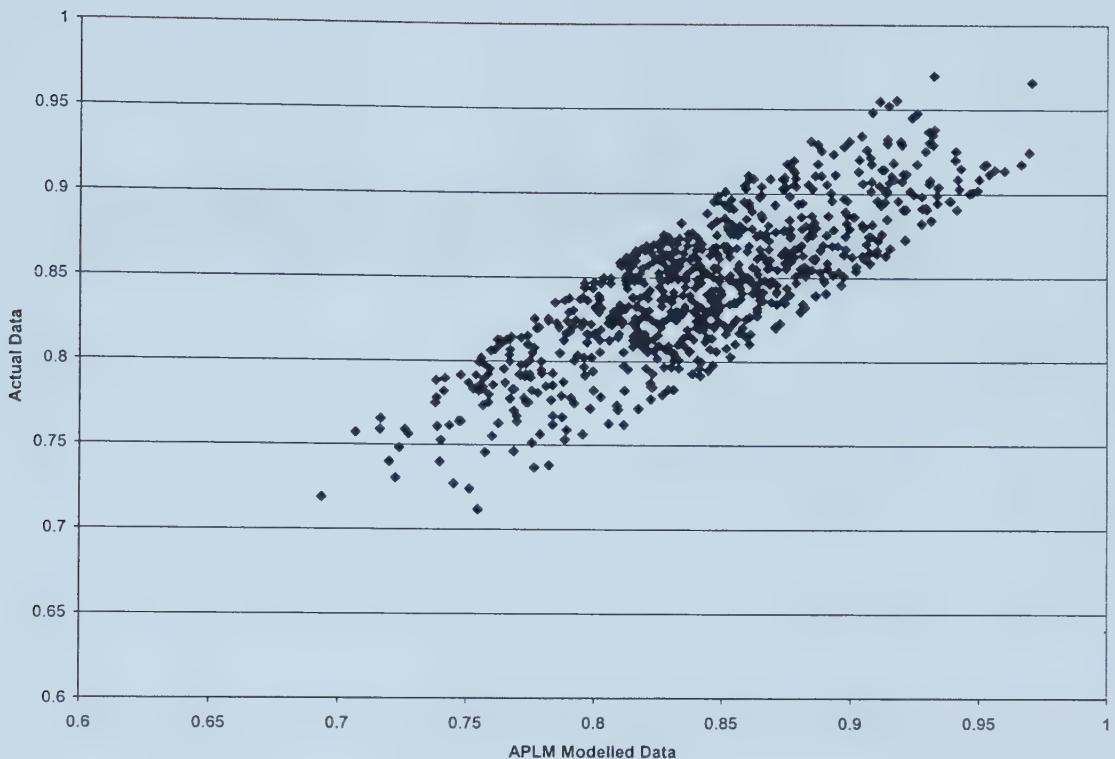


Figure 6.5 Modelled versus Actual Strength Scatter Plot

The models giving the results shown in Figures 6.4 and 6.5 was developed using an end s value of -20. The higher values for s allow the model to perform better at simulating data. However, as in neural networks, the improved curve fitting comes at a cost. In the case of APLMs this cost manifests itself as smaller segments that make analysis difficult. Therefore, the model was rerun with the same data set, with the end value for s set to -12. The results of this new model are shown in Figure 6.6 and 6.7. A comparison of Figure 6.4 and 6.6 illustrate the degradation in simulation capability. The linear parameters for the second model are shown in Table 6.2

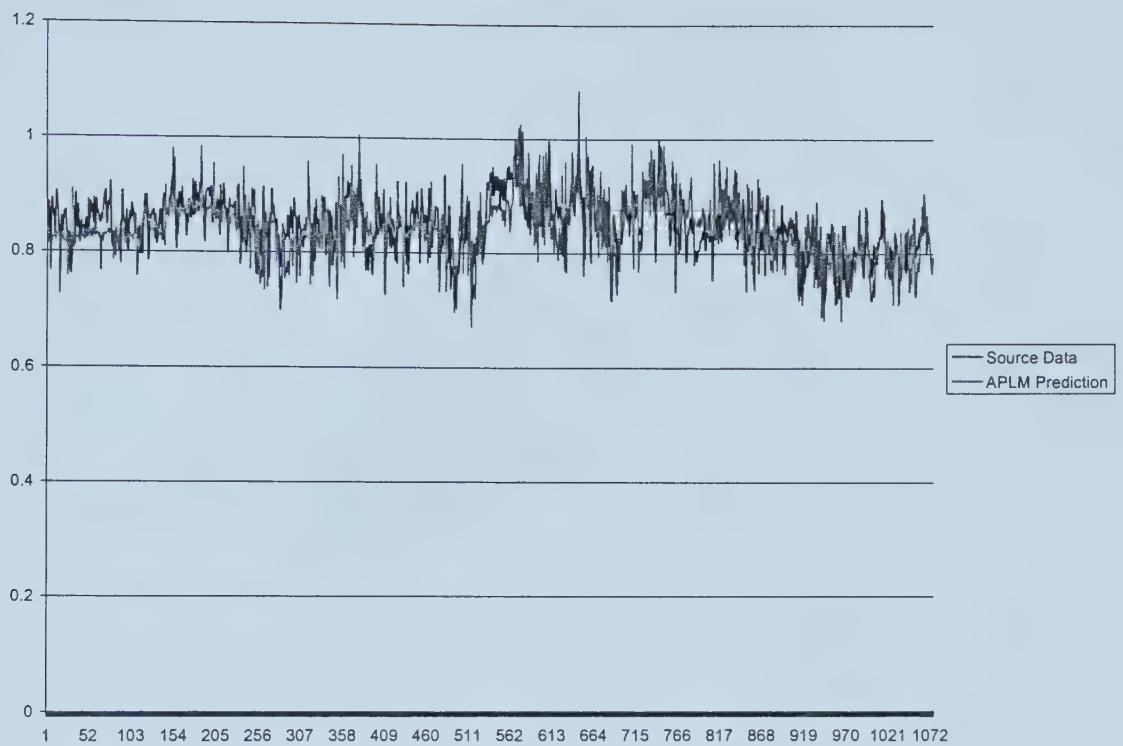


Figure 6.6 APLM Trend Plot for Model 2

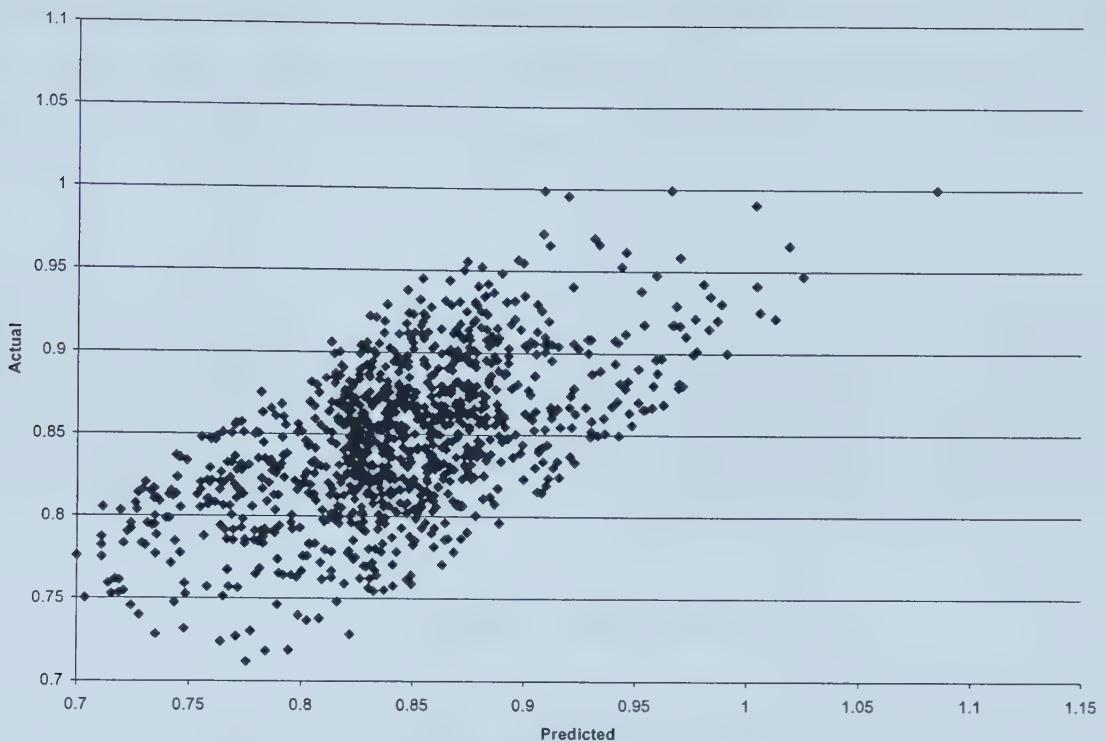


Figure 6.7 APLM Model 2 Scatter Plot for Model 2

	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Pts	Seg	Bias	V	2012	2272	454	2137	FC062	FY062	M	416	1412
2	506	LP1	0.0006	0.0051	0.0027	0.0000	0.0024	0.0008	0.0003	-0.0006	0.0066	0.0019	0.0002
3	258	LP2	-0.0110	-0.0322	0.0201	0.0003	-0.1163	-0.0051	0.0036	-0.0163	0.0188	0.0014	0.0084
4	310	LP3	-0.0110	0.0167	0.0200	-0.0013	-0.1661	-0.0043	0.0102	-0.0161	0.0188	0.0017	0.0081
5	1074	Mean	0.0061	0.0149	0.0118	0.0005	0.0770	0.0029	0.0040	0.0088	0.0131	0.0017	0.0044

Table 6.2 APLM Developed Linear Coefficients for Model 2

The means in the Table 6.2 and 6.3 is a weighted average of the 3 linear segments. A comparison of Table 6.1 and 6.2 show some agreement and some disagreement between the neural network software and the APLM. The two variables that are rated highly by both systems are the variables V and 454. V is a secondary measure of strength, 454 is a measure of the amount of chemicals used in the cook and therefore is a direct indication of how much fibre damage may be expected due to the harsh chemical conditions. Therefore the analysis by the APLM is more consistent with existing process knowledge than that by the neural network.

Because of the differences between the neural network results, the APLM results and knowledge of the process, another APLM model was built replacing many of the variables with ones that made sense from a process-engineering standpoint. The linear coefficients are shown in Table 6.3

	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Pts	Seg	Bias	V	138	47	454	2137	349	247	M	2108	1412
2	36	LP1	0.0064	0.0226	-0.0182	0.0005	0.0886	0.0014	0.0007	-0.0110	0.0074	0.0025	0.0017
3	143	LP2	0.0064	-0.0133	0.0042	0.0005	0.0524	0.0012	0.0001	-0.0110	0.0074	0.0012	0.0014
4	875	LP3	0.0064	0.0012	0.0105	0.0005	-0.0211	0.0001	0.0004	0.0017	0.0074	-0.0003	-0.0005
5	1074	Mean	0.0065	0.0039	0.0100	0.0006	0.0278	0.0005	0.0005	0.0034	0.0075	0.0006	0.0007

Table 6.3 APLM Developed Linear Coefficients for Model 3

A comparison of the slopes or influence caused by the variables is shown in Table 6.4. The approach being used is to use the approximation that the output variable (pulp strength) has a linear dependency on a combination of inputs. The slope with respect to the output variable measures the influence for each variable. Therefore the variables with the greatest (absolute value) slope have the greatest influence. In this table, the influence is normalised as percent of total influence explained by the model with total influence being the algebraic sum of the absolute values of all slopes for a particular segment.. This is necessary when comparing models as a direct comparison on linear coefficients does not adequately indicate the distribution of knowledge in the data set. As more variables contribute knowledge to the model, the variables that in previous models had a certain contribution will have seemed to change. By normalising these contributions it is easier to compare models. In this case only the weighted mean influences are used, however information is also contained in the influence tables or the individual segments

Variable		Percent of variation		dataset 1a model 2	
Model 3	Model 2	Percent of variation	Percent of variation	Model 3	Model 2
454	454	50	56	0.0278	0.077
138		18		0.01	
M	M	14	9	0.0075	0.013
V	V	7	11	0.0039	0.0149
	2012		9		0.0118
247		6		0.0034	
	fy62		6		0.0088
1412	1412	1	3	0.0007	0.0044
2137	2137	1	2	0.0005	0.0029
	fc62		2		0.0029
47		1		0.0006	
2108		1		0.0006	
	416		1		0.0017
349		1		0.0005	
	2272		0		0.0005
		100	100	0.0555	0.1379

Table 6.4 Comparison of APLM Models 2 and 3

Table 6.4 illustrates that the choice of variables by model 3 do indeed provide better explanation of the influence, especially the knowledge added by variable 138 which contributes 18% of the influence. However, not all the additions were beneficial, and the removal of at least one variable was detrimental (2012) with a loss of 9% of the influence. This illustrates that when working with large data sets, an iterative process is required to get to a parsimonious data set that can adequately describe the process.

6.5 GEOMETRIC ANALYSIS OF APLM DATA

In both Models 2 and 3, there is a change in the weights of the variables at different points in the process operating space while some variable's weights are nearly constant throughout. Segment 1, 2, and 3 account for 47%, 24%, and 29% of the data points respectively. Between the segments there are significant changes in the linear coefficients. The changes between segments are consistent with the modelling history of this process. There have been several attempts to model pulp strength using traditional methods but the models always prove to be only valid for a small period of time. After that, something in the process changes and the model is no longer valid. The changes that occur have never been identified. This table

illustrates the three distinct operating ranges in the process that must be modelled separately. There is no single linear model that can accommodate this process.

The angles between two segments a and b can be calculated using the formula

$$\cos(\gamma) = \frac{a \cdot b}{|a||b|} \quad (6.2)$$

giving

Between Segments		Angle (degrees)
1	2	103
1	3	95
2	3	21

Table 6.4 Angles Between Segments

These angles show that there is a significant change in the curvature of the process space and again emphasises that traditional linear methods of identifying this process will not be successful.

In conclusion the small number of segments indicate that the problem has some degree of non-linearity, the angles between segments indicate that the degree of non-linearity is significant near the edges of the area studied. Therefore the geometric analysis indicates that there are three areas of quasi-linear operation connected by high curvature intersections. Therefore the linear analysis performed in the past have likely failed because they did not first partition the data into the quasi-linear segments. Any success that have been seen has likely been due to the one large segment which accounts for ~50% of the data. Finally, this analysis has shown that even if an APLM is not used for a full analysis of a problem, an initial check with an APLM can indicate whether linear or non-linear tools should be used for the final analysis.

6.6 SUMMARY

In this chapter, the use of APLMs to model a real process has been illustrated. Strengths over neural approaches have been discussed. These strengths include the ability to generate geometrical knowledge that in turn provides

- The ability to show variable sensitivity knowledge over the entire operating range as opposed to just at the average operating conditions which is common with neural network approaches.
- The ability to show the degree of non-linearity of the process space is by evaluating the angles between segments,
- The ability to show the impact of the process space's non-linearity by evaluating the number of points in each segment,

The first point is important as traditional neural systems fail to identify those variables that are important away from the midpoint operating range. Non-linearity of process spaces is usually only dealt with when a mathematical model of the space is available. In that case the second derivative can be used to determine curvature of the space. Here as discrete data is being dealt with the second derivative is not available, however, the angle between segments is a viable indication of the degree of non-linearity. Both of these pieces of knowledge were used in modelling pulp strength. One unexpected learning was the low degree of non-linearity present in the system as measured by the number of points per segment. However, it did explain why previously linear regression had moderate success. Clearly the linear regression had captured the largest linear space, but then failed later as the process moved out of that operating region.

The analysis of the Pulp Strength problem provided new insight into the macro dependencies for pulp strength. Cooking conditions (as described by variables 454 and 2012) were the most significant contributors to pulp strength and were ranked 1 and 4 respectively. Variables V and M are known to be indicators of strength and were ranked 2 and 3. The belief that several chemicals impacted strength was confirmed when they were found to be ranked 5, 6, and 7 (variables FY062,1412, & FC062). Variables describing raw material impacts completed the rankings. These results confirmed existing process

engineering beliefs. As well, the variation of the impacts over the operating range could be seen from an analysis of the geometric indicators.

Future work calls for continued refinement of the model and the development of designed experiments to corroborate these findings.

CHAPTER 7

7.0 SUMMARY OF THESIS CONTRIBUTIONS

This research thesis has provided several contributions to the body of knowledge on modelling, using piecewise linear segments, headbox control and pulp strength. As well, a general streamlined MIMO control law has been derived.

7.1 MODELLING

Several modelling methodologies are based on piecewise (linear or otherwise) systems. Many fuzzy systems fall into the piecewise linear category. This research has illustrated some of the issues associated with piecewise modelling and in particular has shown those current linear modelling technologies:

- Rely excessively on visualisation as a knot determination method which in turn limits many applications to simple two dimensional problems,
- Fail to consider the geometry of the process space before building a model. This affects the modellers ability properly specify a data-set, or to select a proper modelling tool,
- Fail to examine the geometry of the model space after it is built to determine the key control aspects of the modelled process,
- Fail to provide multivariate sensitivity analysis across the entire process operating range, and,
- Fail to use fuzzy learning in the development of fuzzy rules (for fuzzy modelling tools).

The Adaptive Piecewise Linear Modelling technology developed here solves these problems. Key to this approach is the provision of geometric data about the model that allows the modeller to develop better understanding of the underlying process. The advantages of this approach were illustrated in the pulp strength research.

Typically when looking at sensitivity of a process to its inputs, modelling technologies simply examine the variation of an output to an input when it is dithered around its average value. This approach is required

because the modelling technologies do not generally incorporate multivariate non-linear sensitivity analysis. The Adaptive Piecewise Linear Model permits the modeller to examine slopes across the range modelled. Not only does the modeller gain knowledge about the sensitivity of the outputs to their inputs, but can observe how these sensitivities change at various points in the operating space.

As well, fuzziness is utilised throughout the entire modelling process - from the basic fuzzy cost function utilised in the linear training algorithm to the fuzzy rules that control the internal workings of the modelling tool itself. No other modelling technology is as dependent on the fuzzy methodology. This research has shown that such a fuzzy based system is possible and effective in solving many problems associated with process engineering.

7.2 MODEL BASED HEADBOX CONTROL

Research on headboxes has generally ignored the pulp machine as the time constants in those systems were felt to be too slow to benefit from advanced model based control schemes. This research has shown the fallacy of that logic. By utilising a high-speed model, an otherwise well tuned coupled PID controller is seen to cause significant overshoot during rate changes. It is shown that a model based predictive controller can provide steady state control at a comparable level to the PID control while at the same time eliminating the overshoot.

Two model based predictive controllers as applied to headbox control were illustrated within this thesis. First, a general MIMO control law was developed using traditional techniques. Then, an APLM based control scheme was developed. It was shown that the APLM had an advantage in that it could transform non-linear processes into segmented linear models. This capability was illustrated on a dynamic dataset. This allows the modeller to create simple model predictive gain-scheduled control schemes that can be easily implemented in a DCS. Other modelling technologies allow non-linear control schemes to be developed, but again, the models often do not indicate to the modeller anything about the underlying process. APLMs with their geometric knowledge provide superior process knowledge.

7.3 PULP STRENGTH

The research into pulp strength provided new insights into the macroscopic contributors to pulp strength in the pulp mill process. The strength of pulp was found to relate to several chemicals utilised in the process, the processing conditions in the digester, and the processing conditions on the machine. It is important to note that the results of the APLM agreed in general with the previous research conducted by this author [Zurcher 1995]. In that case, the knowledge was based on process knowledge and best guesses as to what the neural network was indicating. The data put forward by the APLM was not only consistent with the previous results, but, required less “guesstimating” due to the geometric approach to process knowledge display.

The developed geometric knowledge showed that the process is definitely non-linear. However, the non-linearity was not as significant as expected. It must be noted that previous analyses (both neural networks and others) have not indicated the degree of non-linearity present in this process. Further research is required to understand the best method of taking advantage of this knowledge. One option is to restrict the pulp mill operation to one of the linear ranges which would provide for stable deterministic operation. Another option is to operate at the lowest cost region, which may well take the plant into a non-linear operating region. Further study is required in this area.

7.4 SUMMARY

Several new advances in modelling were developed through this research. A version of model based predictive MIMO control using matrix differentiation techniques was developed. This control was used in a comparison of control schemes – MBC, PID control and an inverted APLM model - as applied to the headbox. It was found that all control schemes offered good control; however, the PID control suffered from significant overshoot during rate changes. The same degree of overshoot was not exhibited by the model based control schemes. Finally, the APLM modelling technology used in developing a static model of pulp strength, highlighted various contributors to pulp strength confirming previous process engineering knowledge while at the same time extending this knowledge by quantifying that knowledge.

Future research is required into the development of more refined pulp strength models and designed experiments to verify the model based findings.

REFERENCES

Allison,B.J., Dumont, G.A., Novak, L.H. Multi-Input Adaptive-Predictive Control of Kamyr Digester Chip Level, PPR780, Paprigan, Pointe Claire, Que. 1990

Allison,B.J., Roberts,C. Generalized Predictive Control of Chip Level in Dual-Vessel Continuous Digesters, MR293, Paprigan, Pointe Claire, Que. 1994

Armstrong W.W, Dwelly, A., Liang, J., Lin, D. Reynolds, S. *Experience Using Adaptive Logic Networks*, Proceedings IASTED Int'l Symposium on Computers, Electronics, Communication and Control, Calgary 1991, pp44-47

Armstrong W.W., Thomas M.M., *Control of a Vehicle Active Suspension Model using Adaptive Logic Networks*, Proceedings World Congress on Neural Networks 1992

Armstrong, W.W., Chu, C. Thmoas, M.M., *Using Adaptive Logic Networks to Predict Machine Failure*, Proceedings Battell Pacific Northwest Labs Workshop on Environmental and Energy Applications of Neural Networks, Richland Wa. 1995

Banchoff, T., Wermer, J., Linear Algebra Through Geometry, Springer, New York, 1991

Brogan, W., Modern Control Theory 3rd Edition, Prentice Hall, Upper Saddle River, New Jersey, 1991

Chessari,C.J.,Barton,G.W.,Watson,P., *On The Development of a Neural Network Based Orthogonal Nonlinear Principal Component Algorithm for Process Data Analysis*, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Cox, E., The Fuzzy Systems Handbook, Academic Press, New York, 1998

Dayal, B.S., *Feedforward Neural Networks for Process Modelling and Control. Master Thesis*, McMaster University, 1992

Diamantaras,K.I., *Robust Principal Component Extracting Neural Networks*, The 1996 IEEE International Conference on Neural Networks, IEEE, 1996

Dierckx, P., Curve and Surface Fitting with Splines, Oxford University Press, Oxford, 1993

Egri, P.A., Underwood, P.F., *HILDA: Knowledge Extraction from Neural Networks in Legal Rule Based and Case Based Reasoning*, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Friedman, J.H., *Multivariate Adaptive Regression Splines*, Annals of Statistics, Volume 19, Mar 1991, pp1-67, Institute of Mathematical Statistics

Goodman, R.M., Miller, J.W., Smyth, P., *An Information Theoretic Approach To Rule-Based Connectionist Expert Systems*, Advances in Neural Information Processing Systems I, Touretzky, D.S. editor, Morgan Kaufmann, San Mateo Ca,

Ham, F.M., Collins, E.G., *A Neurocomputing Approach for Solving the Algebraic Matrix Riccati Equation*, The 1996 IEEE International Conference on Neural Networks, IEEE, 1996

Hashem, S., Schmeiser, B., Improving Model Accuracy Using Optimal Linear Combinations of Trained Neural Networks, IEEE Transactions on Fuzzy Systems, IEEE, May 1995

Hayashi, Y., *A Neural Expert System with Automated Extraction of Fuzzy IF-THEN rules and its Application to Medical Diagnosis*, Advances in Neural Information Processing Systems 3, Lippman, R.P., Moody, J.E., Toureszky, D.S., Editors, Morgan-Kaufman, San Mateo, 1989

Hecht-Nielsen, R., Neurocomputing, Adison-Wesley, New York, 1989

Jang, J.-S.R., Sun, C.-T., Muzutani, E., Neuro-Fuzzy and Soft Computing, Prentice-Hall, Upper Saddle River, NJ. 1997

Kim, H., Shen, X., Rao, M., Zurcher, U.J., *Quality Prediction by Neural Network for Pulp and Paper Processes*, Canadian Electrical and Computer Engineering Conference, Vancouver Sept 14-17, 1993, pp104-107

Kosko, B., Neural Networks and Fuzzy Systems, Prentice Hall Englewood Cliffs, New Jersey, 1992

Kreyszig, E., Advanced Engineering Mathematics, John Wiley and Sons, Inc., New York, 1993

Lane, M., Headbox Decoupling Strategy, Weyerhaeuser correspondance, 1991

Lebeau, B., Arrese, R., Bauduin, S., Grobet, R., Foulard, C., Non Interacting Multivariable Paper Machine Headbox Control: Some Comparisons with Classical Loops, Instrumentation and Automation in the Paper, Rubber, Plastics and Polymerisation Industries, IFAC, 1980

LeCun, Y., Denker, J.S., *Transforming Neural-Net Output Levels to Probability Distributions*, Advances in Neural Information Processing Systems 3, Lippman, R.P., Moody, J.E., Touretzky, D.S., editors, Morgan Kaufmann, San Mateo Ca,

Li,I., Kwok, E., Zurcher, U.J.*Prediction and Prevention of Sheetbreaks using PLS and an Expert System*, IFAC Safe Process International Symposium on Fault detection supervision and safety for technical process, Hull, UK, 1997,pp1165-1170

Meher, P.K., Pradhan, S.K., *Optimal Neural Network for Solution of Lyapunov Equation*, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Minsky, Papert, S, Perceptrons, MIT Press, 1969.

Miyoshi, T., Nakao, K., Ichihashi, H., Nagasaka, K., *Neuro-Fuzzy Pursuit Regression*, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Nader, A., Dual Model Reference Adaptive Control on Paper Machine Headboxes, Instrumentation and Automation in the Paper, Rubber, Plastics and Polymerisation Industries, IFAC, 1980

Nie, J., Lee, T.H., *Building Fuzzy Systems by Soft Competitive Learning*, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Nie, J., Linkens, D.A., *Learning Control Using Fuzzified Self-Organizing Radial Basis Function Network*, IEEE Transactions on Fuzzy Systems, IEEE, November 1993

Oja,E., *A Simplified Neuron Model as a Principal Component Analyzer*, Journal Mathematical Biology, vol15, 1982

Passino, K.M., Yurkovich, S., Fuzzy Control, Addison-Wesley, New York, 1997

Parker, D.B., *A comparison of algorithms for neuron-like cells*, Proc. Second Annual Conf. on Neural Networks for Computing Denker, J. (Editor), 151 327-332, AM. Inst. of Physics, New York, 1986

Peh, L., Tsang, C., Information Measure of Knowledge Extracted from Neurons as a Tool for Analyzing Boolean Learning in Artificial Neural Networks, 1995 IEEE International Conference on Neural Networks Proceedings, IEEE, 1995

Petrus,C.J., Allison,B.J. On the Application of Generalized Predictive Control to Digester Chip Level, MR193, Paprican, Pointe Claire, Que. 1990

Rao, M, Qiu, H, Process Control Engineering, Gordon and Breach Science Publishers, Langhorne Pa. 1993

Seborg, D.E., Edgar, T.F., Mellichamp, D.A., Process Dynamics and Control, Wiley, New York, 1989

Sell, N.J,(Editor) Process Control Fundamentals for the Pulp and Paper Industry, TAPPI Press, Atlanta, 1995

Simoff, S.J., *Handling uncertainty in neural networks: An Interval Approach*, The 1996 IEEE International Conference on Neural Networks, IEEE, 1996

Smook G.A., Handbook for Pulp & Paper Technologists, Joint Executive Committee of the Vocational Education Committees of the Pulp and Paper Industry, 1989

Soeterboek, R. Predictive Control, A Unified Approach, Prentice Hall, Herts UK, 1992

Supynuk,A.G., Armstrong W.W., *Adaptive Logic Networks and Robot Control*, Prceedings Vision Interface Conference, Vancouver, 1992, pp181-186

Tawel, R., *Does the Neuron "Learn" like the Synapse?*, Advances in Neural Information Processing Systems I, Touretzky, D.S. editor, Morgan Kaufmann, San Mateo Ca,

Xia, Q., Zurcher, U.J., Rao, M., *Artificial Neural Network Model for a Pulp Machine*, Fall 1994 Western Branch Meeting, CPPA, Jaspar, Paper 2 Section 2

Xia, Q., Rao, M., Shen, X., Ying, Y., Zurcher, U.J., *Systematic Modeling and Decoupling Design and Control of a Pressurized Headbox*, Canadian Electrical and Computer Engineering Conference, Vancouver Sept 14-17, 1993, pp 962-965

Zeng, X.-J., Madan, G.S., *Decomposition Property of Fuzzy Systems and Its Applications*, IEEE Transactions on Fuzzy Systems, IEEE, May 1996

Zhu, J., Xia, Q., Rao, M., Zurcher, U.J., *An Artificial Neural Network Integrated Environment for Modeling and Simulation in Pulp and Paper Industry*, 82nd Annual Meeting, Technical Section, CPPA, 1996

Zhu, J., Xia, Q., Zurcher, U.J., *An Artificial Neural Network Integrated Environment for Modeling and Simulation in Pulp and Paper Industry*, 82nd Annual Meeting, Technical Section, CPPA, 1996

Zhu, J., Zurcher, U.J., *Neural Network Model of a Lagoon*, CPPA Journal, Accepted, 1997

Zurcher U.J., *Adaptive Logic Networks: A New Approach to Process Modelling*, Proceedings CPPA Technical Section Western Branch Fall Conference, 1994

Zurcher U.J., Application of AI to Pulp Mill Operations, Masters of Science Thesis, University of Alberta, 1995

Zurcher, J.G., Zurcher, U.J., *Computer-Human Cybernetics and On-line Documentation*, 1997 IEEE Conference on Man Machiner & Cybernetics

Zurcher, U.J., *Adaptive Logic Networks, A New Approach to Process Modelling*, Fall 1994 Western Branch Meeting, CPPA, Jaspar, Paper 1 Section 2

Zurcher, U.J., Li, P., Kwok, E. *Multivariable Predictive Control for a Pressure/Vacuum Headbox*, CPPA Western Branch Meeting – Jaspar May 15-18 1996

Zurcher, U.J., *Putting Artificial Intelligence to Work*, IEEE Conference on Control Applications, Vancouver Sept 13-16, 1993, pp 687-689

Zurcher, U.J., *Survey of Real Time Expert Systems: An Industry Perspective*, 2nd Intelligence Engineering Laboratory Research Review Conference Aug 24-25 1995, Pp 71-75

Zurcher, U.J., *Weyerhaeuser Canada ISO 9000 Certification*, 1998 META Conference "When Worlds Collide: Preparing for the 21st Century Intranet", Paper 2, Case Study Section

University of Alberta Library



0 1620 2074 2654

B45496